Tom Clancy's Ghost Recon
Igor User Guide
(Sections 6-8)
V1.0

Matthias Dohmen
9/30/2002

Table of Contents

6: In-Depth Single Player Scripting

The basic functions of Igor make it possible to quickly populate a mission with different Elements, give them something to do from the beginning, and let their AI produce interesting behaviors. It's not too difficult to create and use Objectives, or to fill out the Briefing, Map, and Mission properties. In fact, there's quite a bit that can be done with just these basic functions. So what's left?

The rest of the Scripting Language can be used to take a mission above and beyond the expectations of the creator and the users. There are very few things that Igor's Scripting Language cannot do in a mission. Say you wanted to assign the same plan to an entire Platoon – you can set up a loop in the script to handle that quickly and efficiently. What about setting up more dynamic engagements? Having a cinematic push a story forward in the middle of the mission? Explosions? These things and much, much more are at your fingertips when you delve into the Scripting Language. This section will cover as much as we can put down on paper, but there's always new ways to use something as open-ended and powerful as Igor, so feel free to take the ideas and examples presented in this section and run with them.

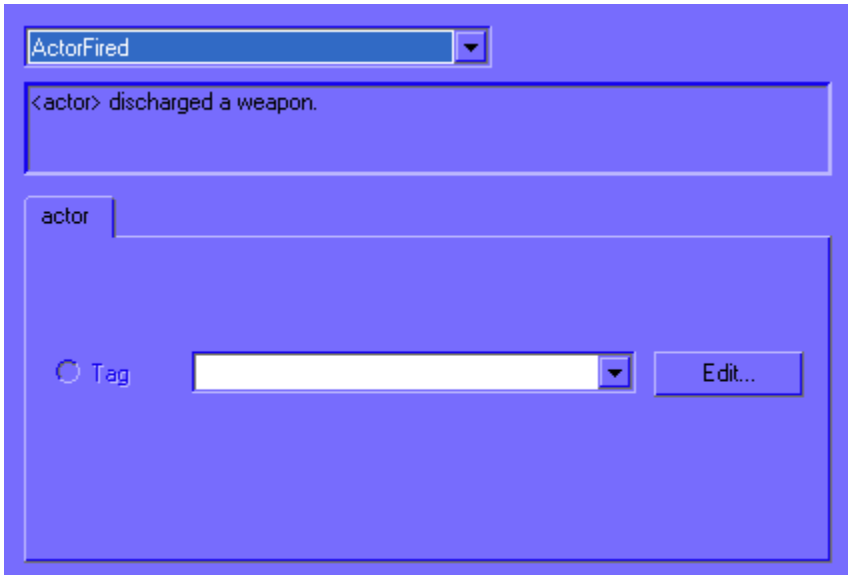
The first part of this section deals with the logic and methodology you should keep in mind while scripting a single player mission. The next part deals with every Trigger available in the Scripting Language. The third part does the same with Responses. The fourth section deals with Parameters, with a list of the different Parameters and what they entail, including Queries. The section after that is a tutorial, putting to use everything you've learned about single-player scripting to create an entire mission. The fifth section covers scripting Special Effects, and the sixth section covers creation of insertion and extraction cinematics as seen in Island Thunder. The seventh section covers the Tools Menu in the Menu Bar, describing and providing examples of all the special Tools in Igor and how they impact a mission. The final section is a full-mission tutorial, covering a large amount of what Igor can do.

Trigger Events

This section covers all the Trigger Events in Igor, provides a description of their function, and examples of their use where appropriate.

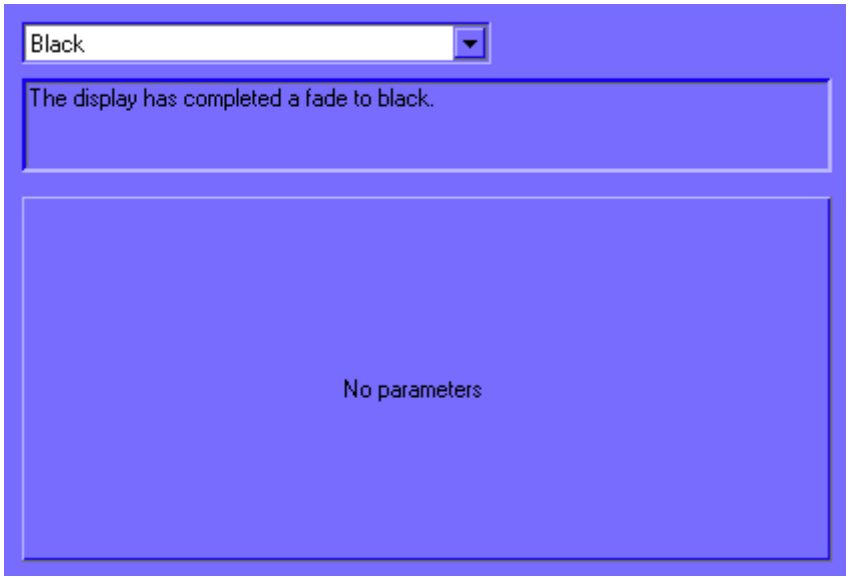
Note that no Parameters can be Queried in a Trigger, but a Reference can be created elsewhere in the script that is Queried using the "VariableSet…" Responses. This is useful if the user wants to trigger a block off of something a Player-controlled Element does, like crossing into an Extraction Zone. Queries are covered in detail in part four of this section, and the "VariableSet…" Responses are covered in part three.

ActorFired



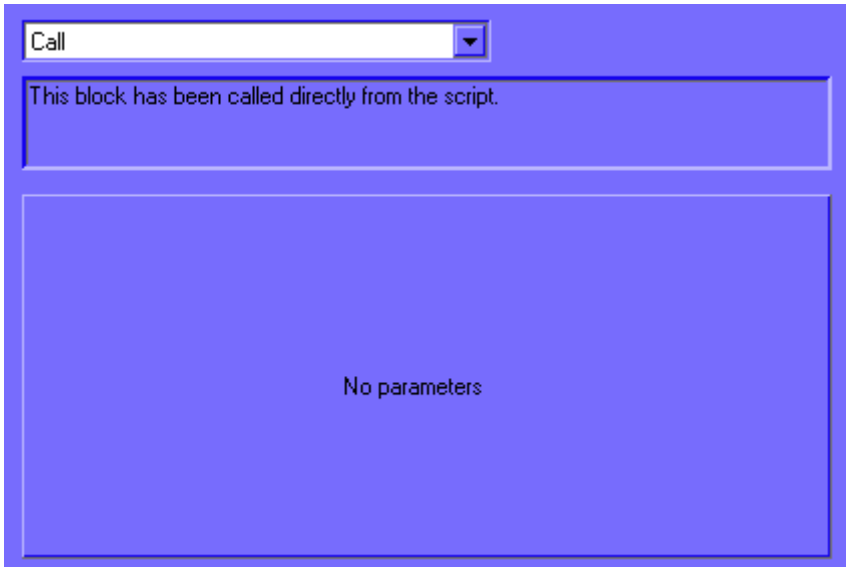
This Trigger activates when the designated Actor fires any weapon that it has been assigned. The only Parameter for this Trigger is the Actor.

Black



This Trigger activates when the display has faded completely to black. It has no parameters and is used only for cinematic purposes.

Call

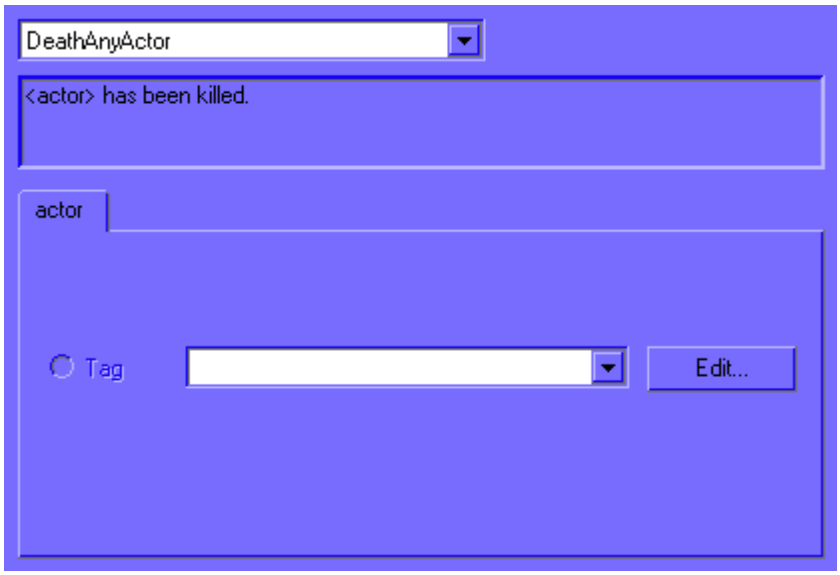


This Trigger activates when its Group is called from a Response (QueueCall or RedirectIf)) in another Block, and is usually part of a Loop. It has no Parameters. It should be the only Block in its Group, and it cannot be part of the Default Group. Its Group does not need to be Enabled for it to work (see "GroupEnable" Response).

DeathActor

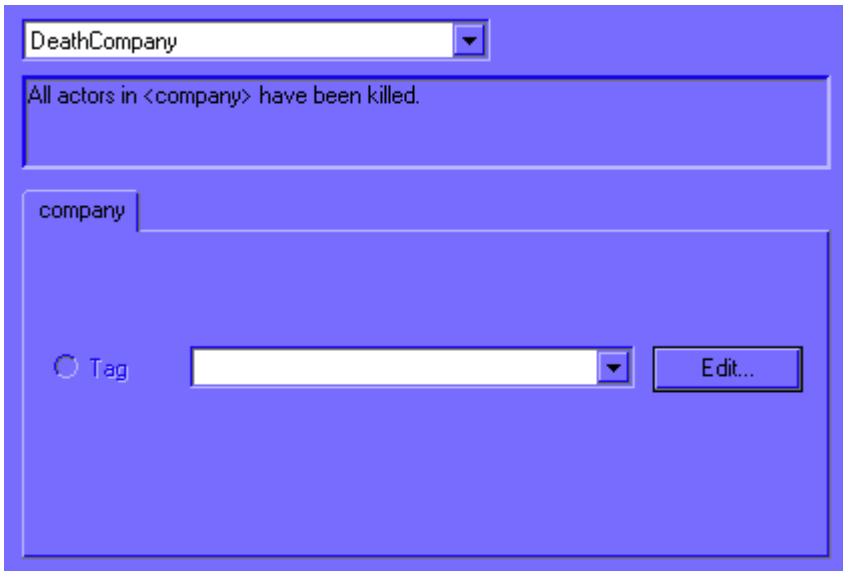
This Trigger activates when the designated Actor has died. Its only Parameter is the Actor.

DeathAnyActor



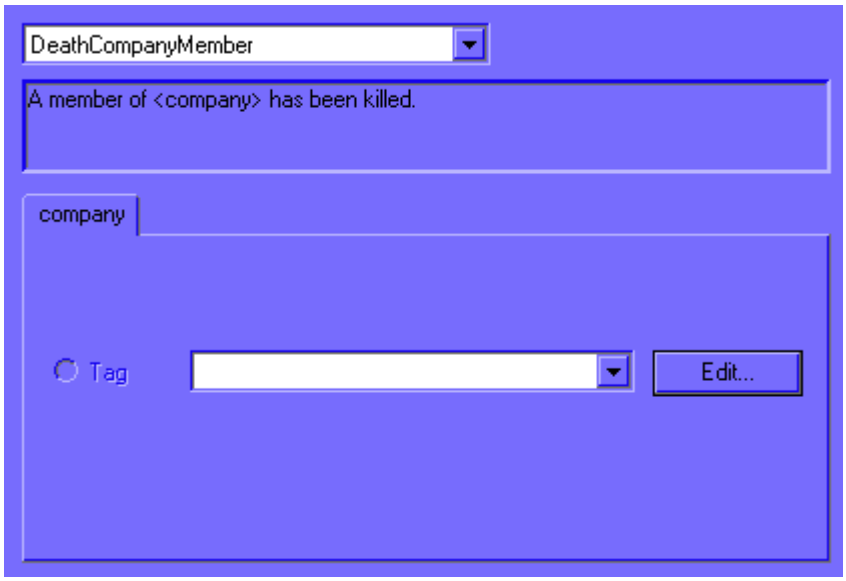
This Trigger activates when any Actor has died, but an Actor Reference must be created for use in its Parameter (i.e. it shouldn't be used with a Placed Actor). Once any Actor has died, that Actor is used for the Actor Reference and may be used anywhere else in the Script, although this is mostly useful in creating new Game Types. Its only Parameter is Actor. See "In-Depth Multiplayer Scripting" section for more details.

DeathCompany



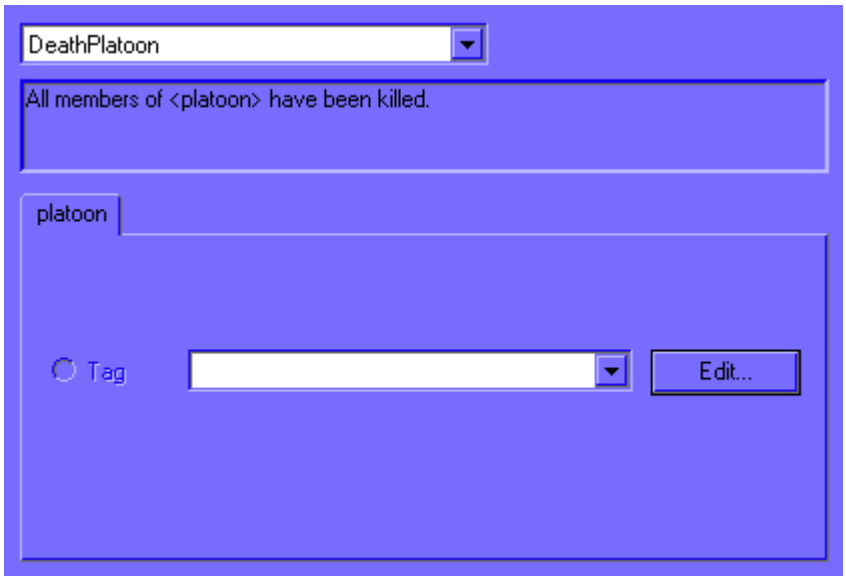
This Trigger activates when all Actors in a Company have died. The only Parameter is Company. Note that if anyone on the designated Company has surrendered, this Trigger will not be activated. Use the "Rout" Triggers to account for surrendering.

DeathCompanyMember



This Trigger activates when an Actor in the designated Company dies. The only Parameter is Company. Note that if an Actor on the designated Company has surrendered, this Trigger will not be activated for that Actor. Use the "Rout" Triggers to account for surrendering.

DeathPlatoon

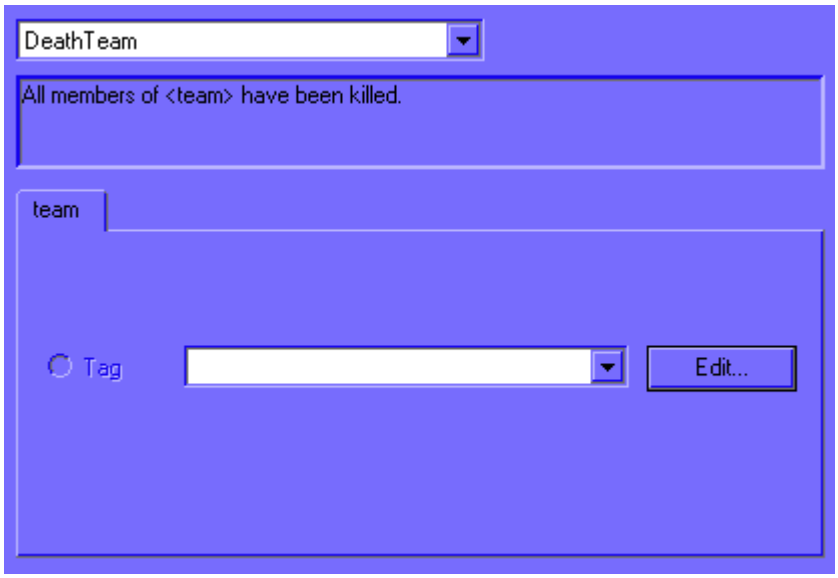


This Trigger activates when all Actors in a Platoon have died. The only Parameter is Platoon. Note that if anyone on the designated Platoon has surrendered, this Trigger will not be activated. Use the "Rout" Triggers to account for surrendering.

DeathPlatoonMember

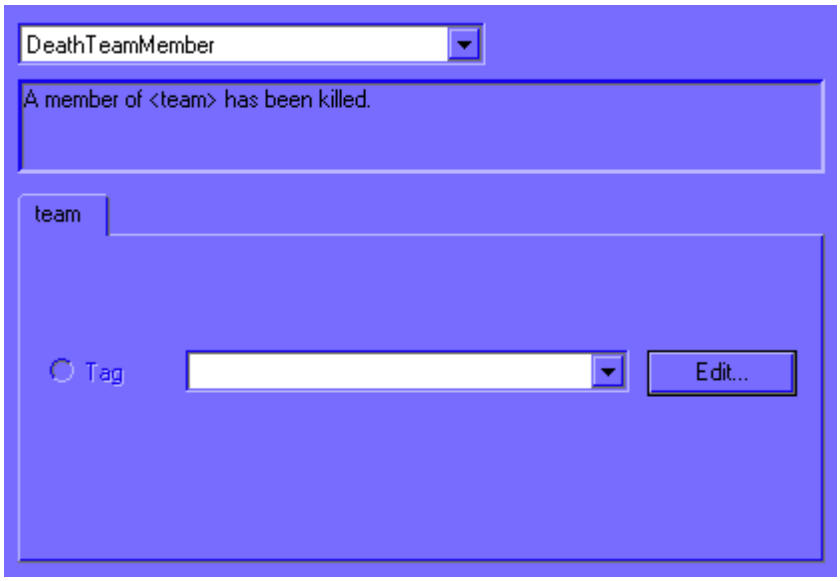
This Trigger activates when an Actor in the designated Platoon dies. The only Parameter is Platoon. Note that if an Actor on the designated Platoon has surrendered, this Trigger will not be activated for that Actor. Use the "Rout" Triggers to account for surrendering.

DeathTeam



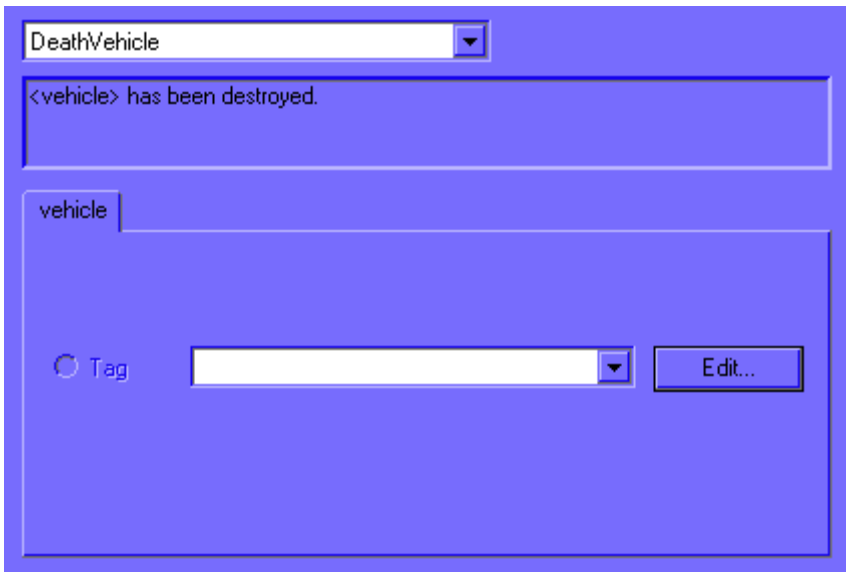
This Trigger activates when all Actors in a Team have died. The only Parameter is Team. Note that if anyone on the designated Team has surrendered, this Trigger will not be activated. Use the "Rout" Triggers to account for surrendering.

DeathTeamMember



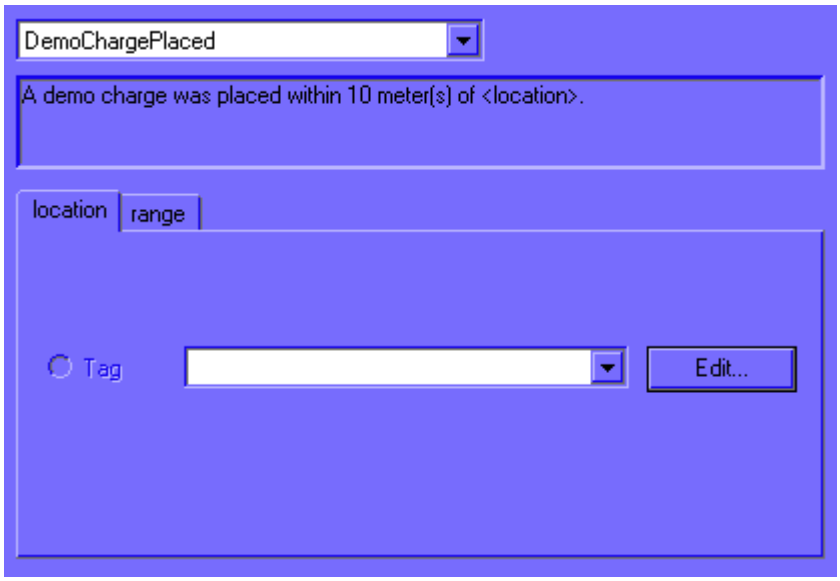
This Trigger activates when an Actor in the designated Team dies. The only Parameter is Team. Note that if an Actor on the designated Team has surrendered, this Trigger will not be activated for that Actor. Use the "Rout" Triggers to account for surrendering.

DeathVehicle



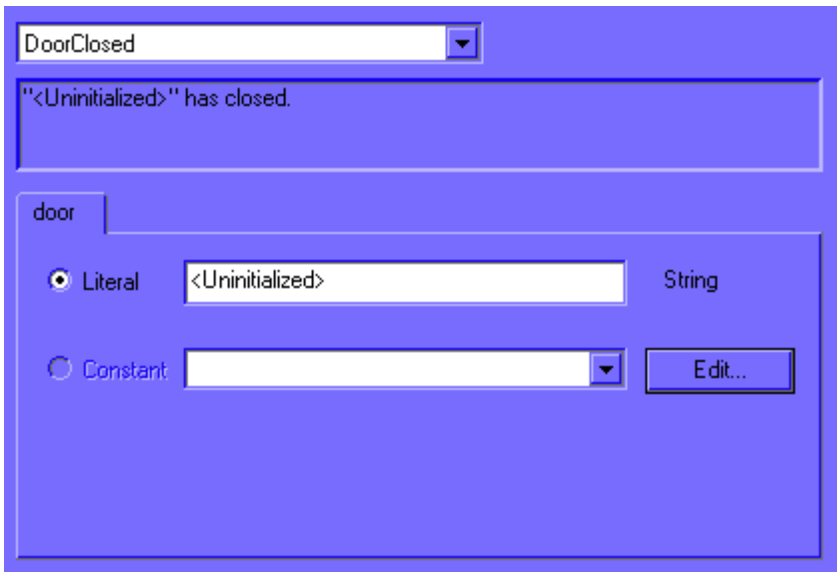
This Trigger activates when the designated Vehicle is destroyed. The only Parameter is Vehicle. Note that DeathCompany and DeathCompanyMember do not include Vehicles in that Company. Also note that the "DestroyVehicle" Response will not activate this Trigger.

DemoChargePlaced



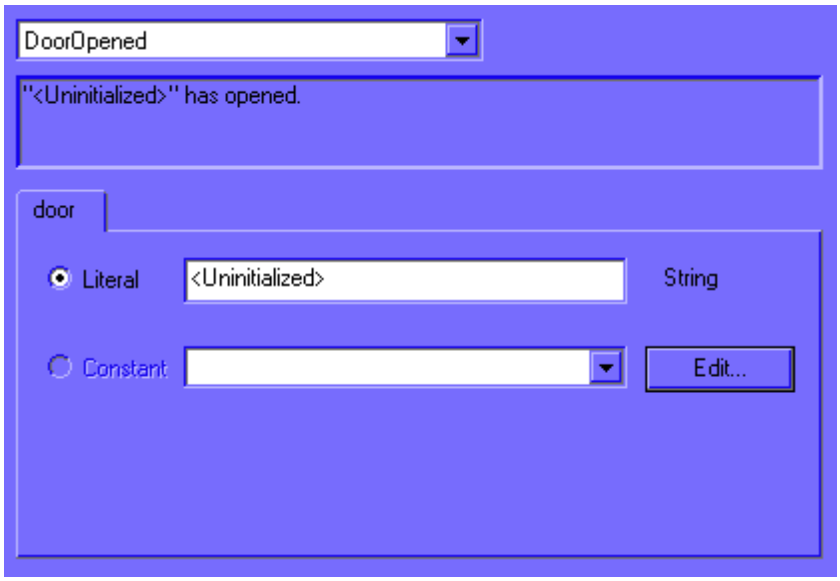
This Trigger activates when a Demo Charge is placed within a set range of a designated location. The Parameters are Location and Range.

DoorClosed



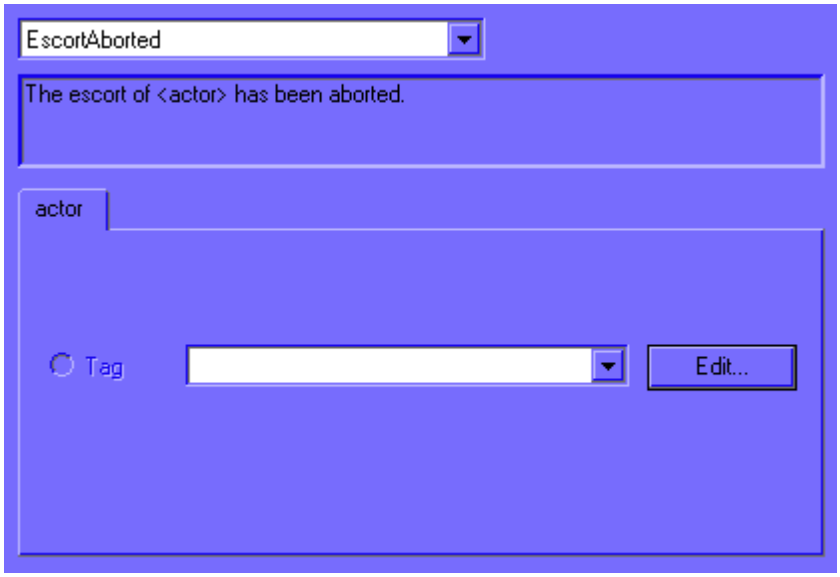
This Trigger activates when the designated Door is closed. The only parameter is Door.

DoorOpened



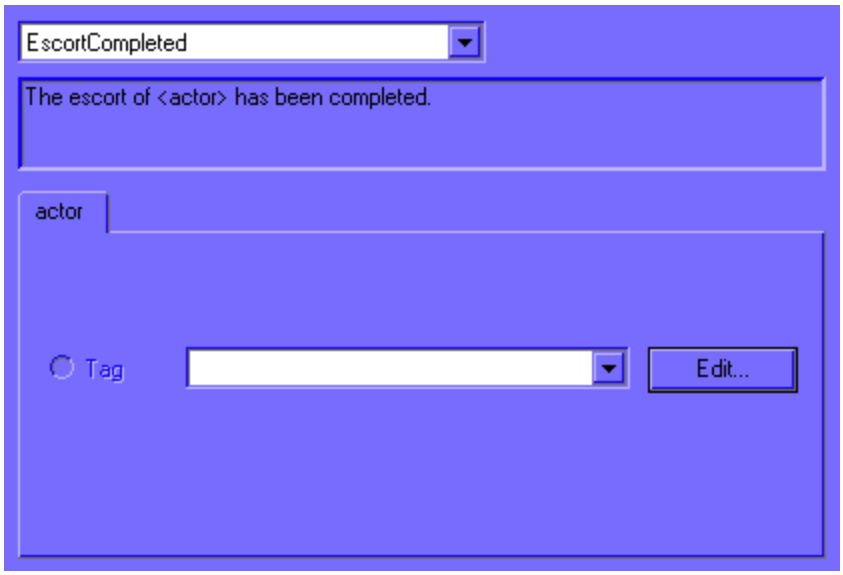
This Trigger activates when the designated Door is opened. The only Parameter is Door.

EscortAborted



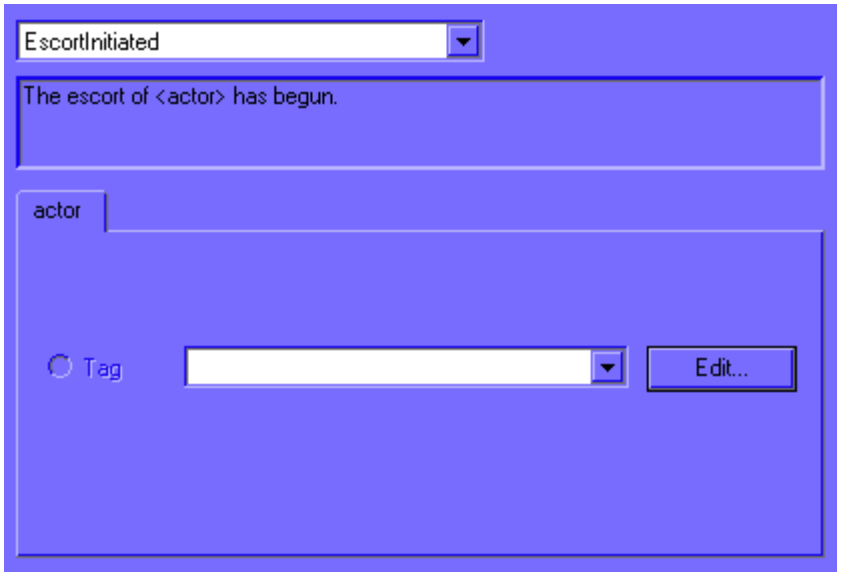
This Trigger activates when the escort of a designated Actor, using the Hostage or Captive behavior, stops. The only Parameter is Actor.

EscortCompleted



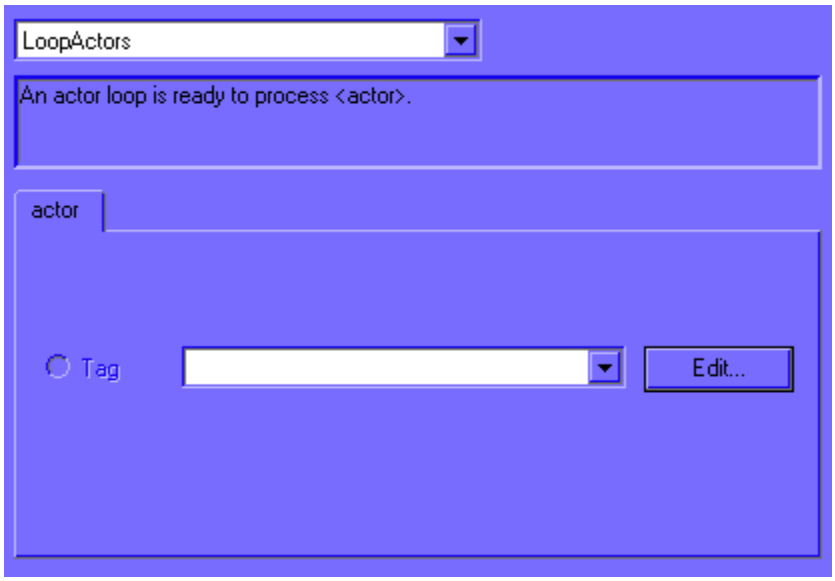
This Trigger activates when the designated Actor using the Hostage behavior reaches any Zone labeled as Extraction in its Zone Properties. The only Parameter is Actor.

EscortInitiated



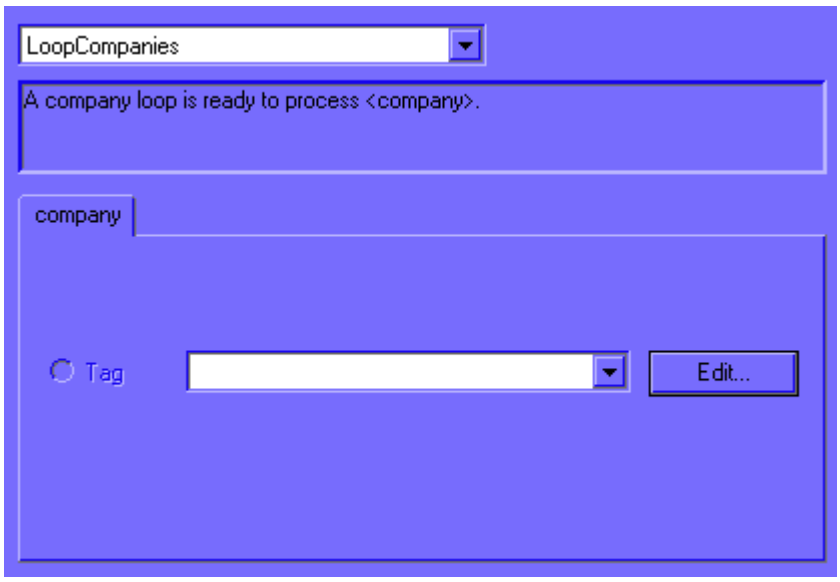
This Trigger Activates when the designated Actor is first escorted. The Actor must be set as a Hostage or a Captive for escort to begin. The only Parameter is Actor.

LoopActors



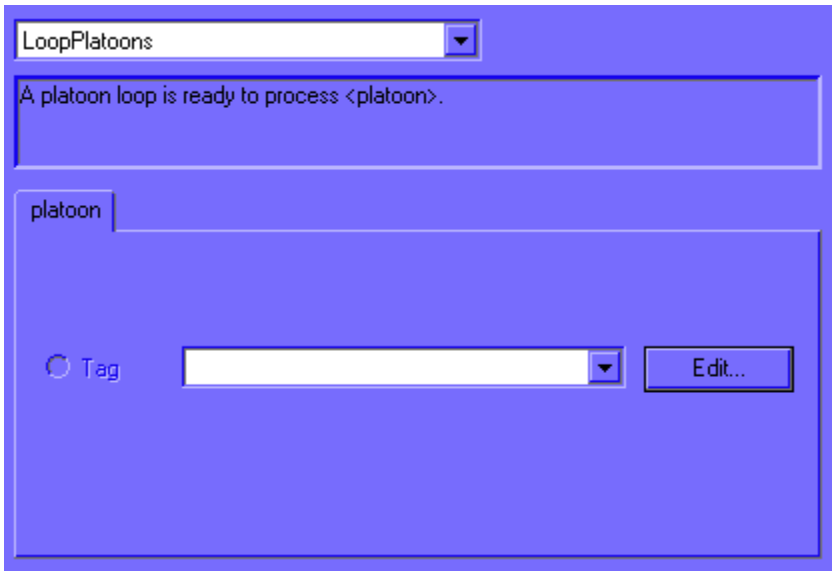
This Trigger activates when its Group is directly called from the script using the "QueueLoop…" Responses. Its only Parameter is Actor. See "In-Depth Multiplayer Scripting" for more information.

LoopCompanies



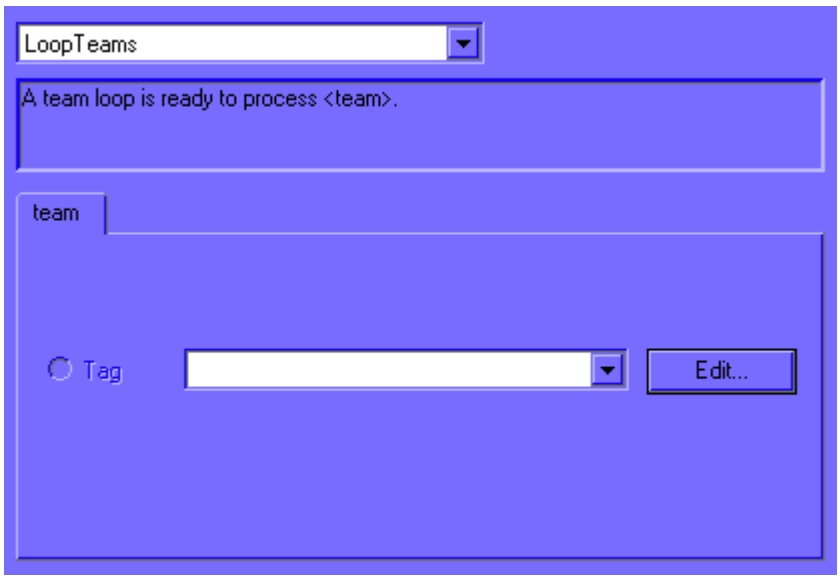
This Trigger activates when its Group is directly called from the script using the "QueueLoop…" Responses. Its only Parameter is Company. See "In-Depth Multiplayer Scripting" for more information.

LoopPlatoons



This Trigger activates when its Group is directly called from the script using the "QueueLoop…" Responses. Its only Parameter is Platoon. See "In-Depth Multiplayer Scripting" for more information.

LoopTeams



This Trigger activates when its Group is directly called from the script using the "QueueLoop…" Responses. Its only Parameter is Team. See "In-Depth Multiplayer Scripting" for more information.

MapObjectDestroyed

This Trigger is activated when the designated Map Object has been destroyed. The only Parameter is Object. Note that when using a literal, it must be typed in exactly as the name of the Map Object appears in the 2d Command Map window when "Map Objects" is selected from the View menu.

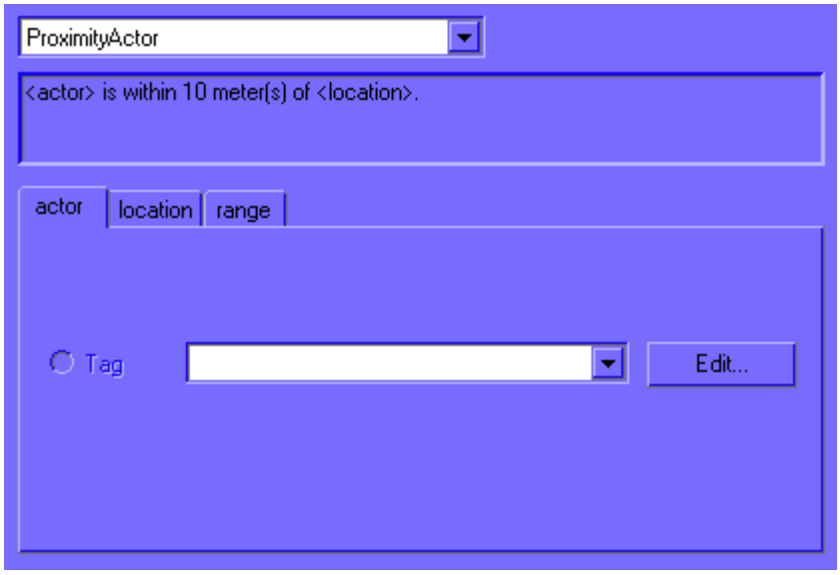
NoPlayersLeft

This Trigger activates when there are no players left alive. It has no Parameters.

PreAction

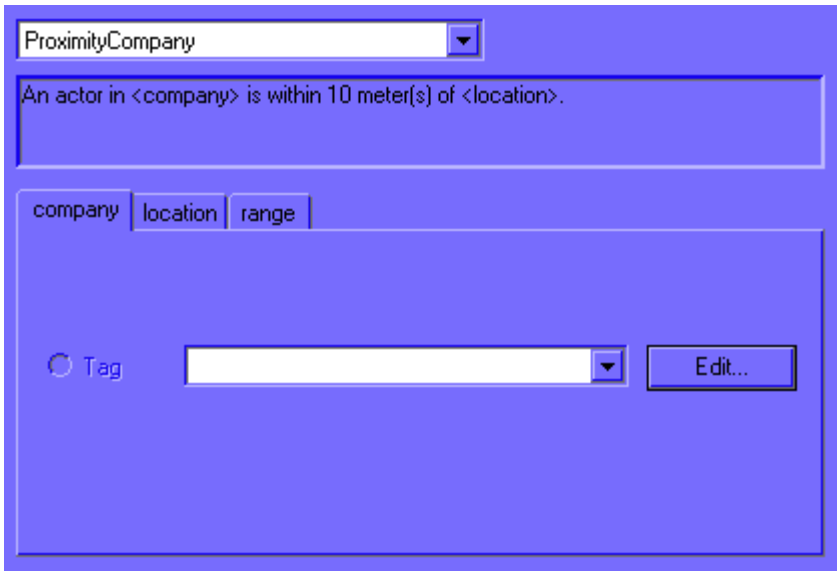
This Trigger activates before the simulation begins. It is used only in special cases, such as hiding Elements, since not all parts of the script have been activated at this point. It has no Parameters.

ProximityActor



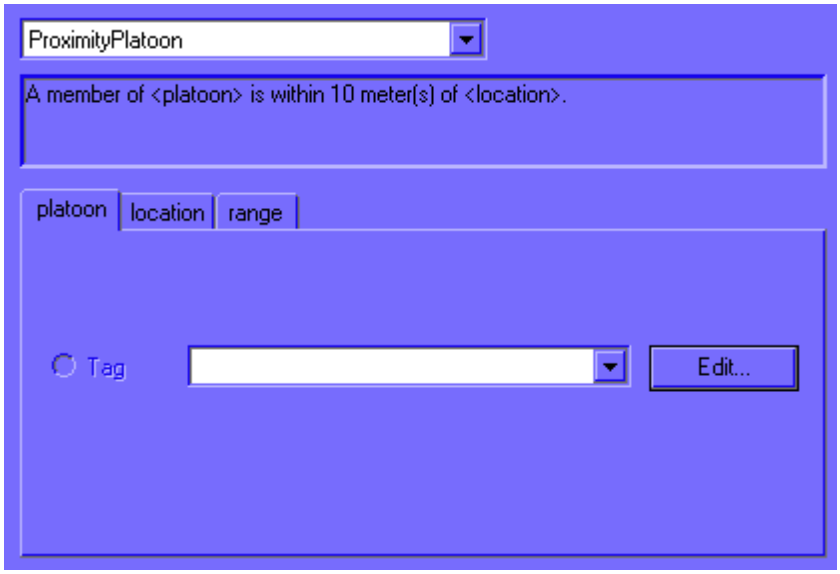
This Trigger activates when the designated Actor is within a set distance of the designated location. If the location is a Zone, the range becomes the edge of the Zone, and it only checks for proximity to the Zone on that Zone's planning level. The Parameters are Actor, Location, and Range.

ProximityCompany



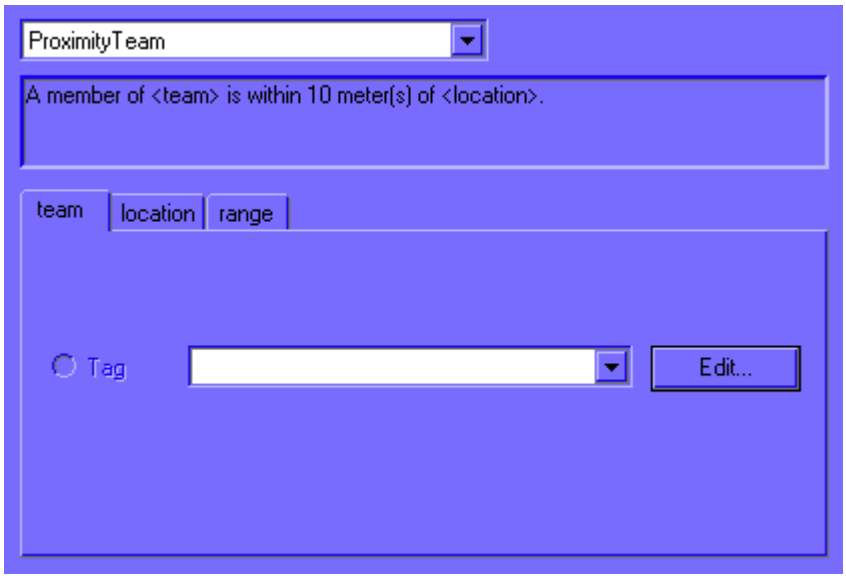
This Trigger activates when any Actor on the designated Company is within a set distance to a location. If the location is a Zone, the range becomes the edge of the Zone, and it only checks for proximity to the Zone on that Zone's planning level. The Parameters are Company, Location, and Range.

ProximityPlatoon



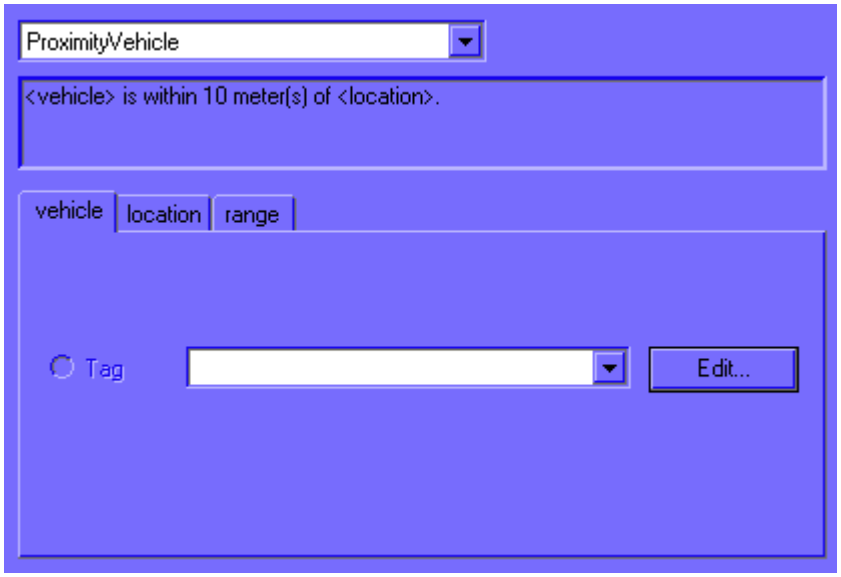
This Trigger activates when any Actor on the designated Platoon is within a set distance to a location. If the location is a Zone, the range becomes the edge of the Zone, and it only checks for proximity to the Zone on that Zone's planning level. The Parameters are Platoon, Location, and Range.

ProximityTeam



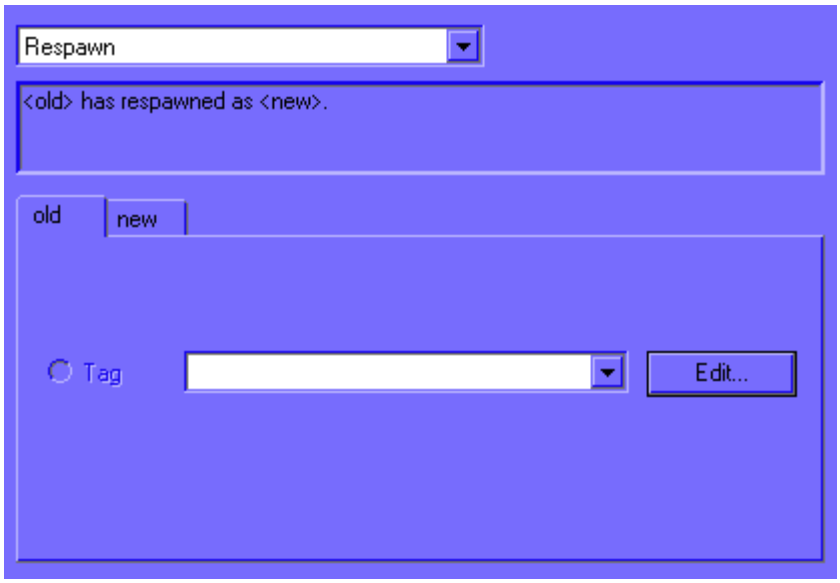
This Trigger activates when any Actor on the designated Team is within a set distance to a location. If the location is a Zone, the range becomes the edge of the Zone, and it only checks for proximity to the Zone on that Zone's planning level. The Parameters are Team, Location, and Range.

ProximityVehicle



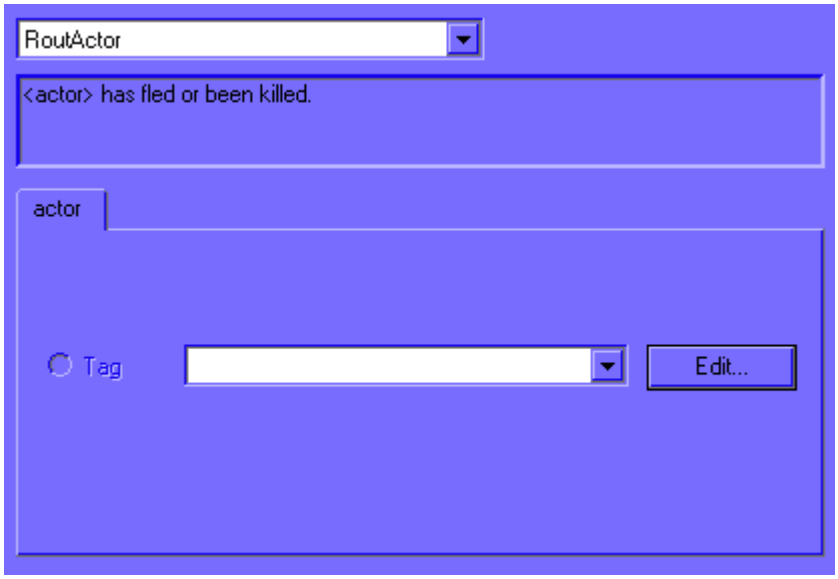
This Trigger activates when the designated Vehicle is within a set distance to a location. If the location is a Zone, the range becomes the edge of the Zone, and it only checks for proximity to the zone on that Zone's planning level. The Parameters are Vehicle, Location, and Range.

Respawn



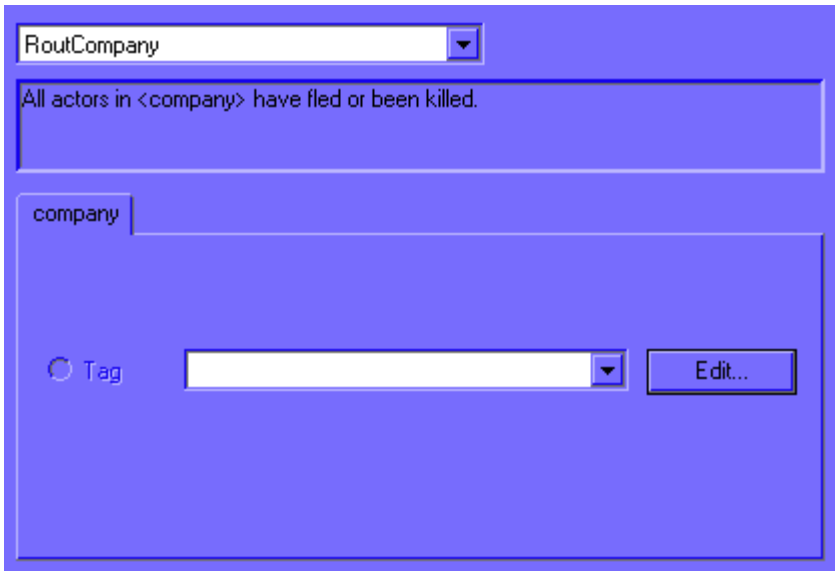
This Trigger activates when the designated Actor has respawned and need a new reference. The Parameters are Old and New. See "In-Depth Multiplayer Scripting" for more information.

RoutActor



This Trigger activates when the designated Actor surrenders or is killed. The only Parameter is Actor.

RoutCompany

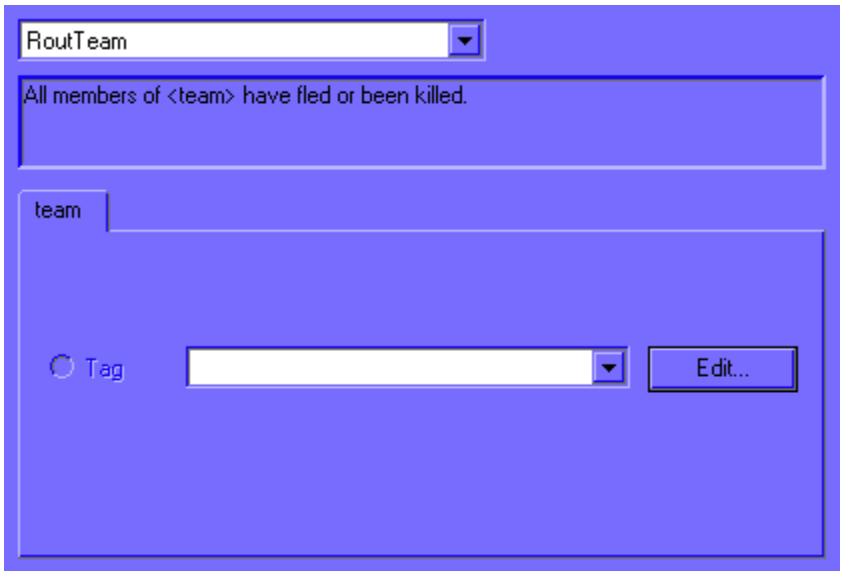


This Trigger activates when all the Actors in the designated Company surrender or are killed. The only Parameter is Company.

RoutPlatoon

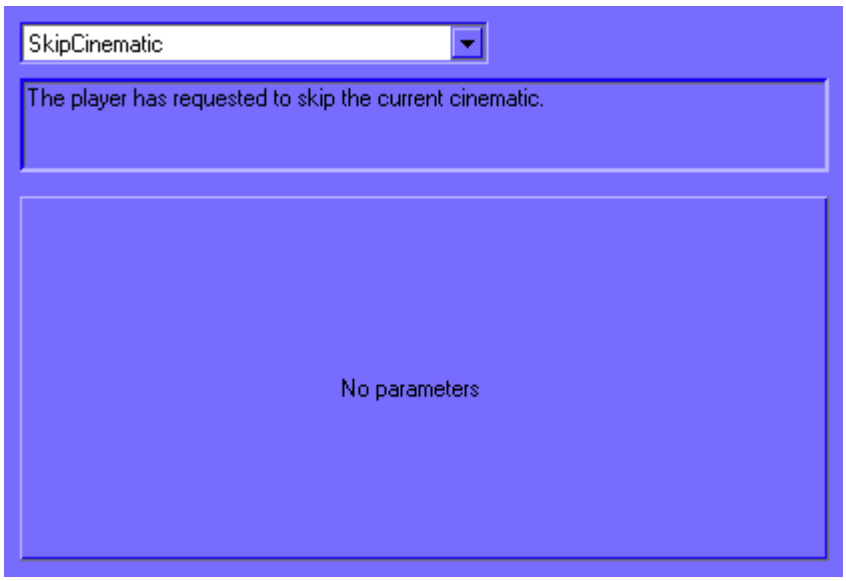
This Trigger activates when all the Actors in the designated Platoon surrender or are killed. The only Parameter is Platoon.

RoutTeam



This Trigger activates when all the Actors in the designated Team surrender or are killed. The only Parameter is Team.

SkipCinematic

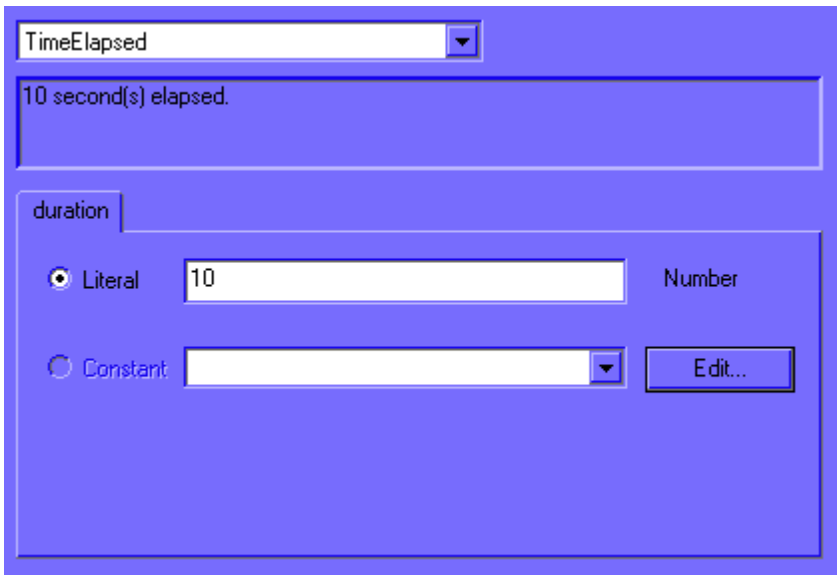


This Trigger activates when the player presses the spacebar or esc keys on the keyboard while the game is in cinematic mode. There are no Parameters.

Startup

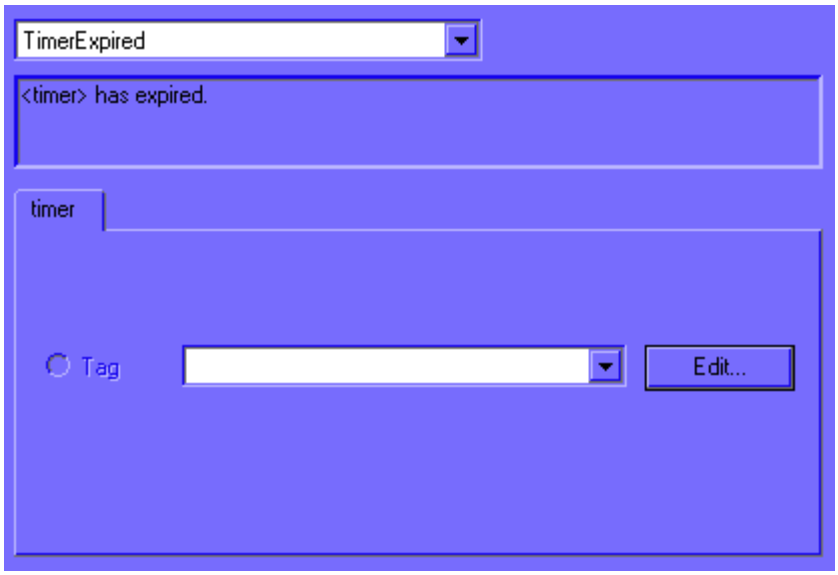
This Trigger activates when the simulation is starting, i.e. when the action phase of the game begins. It has no Parameters.

TimeElapsed



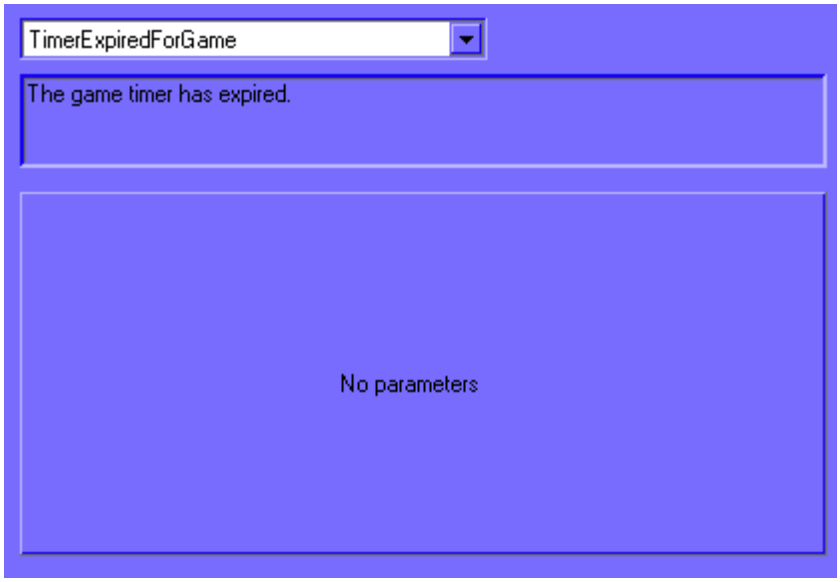
This Trigger activates when the specified amount of time has passed since its Group was enabled. If the Group is Default, the amount of time elapsed is calculated from the start of the simulation.
The only Parameter is Duration.

TimerExpired



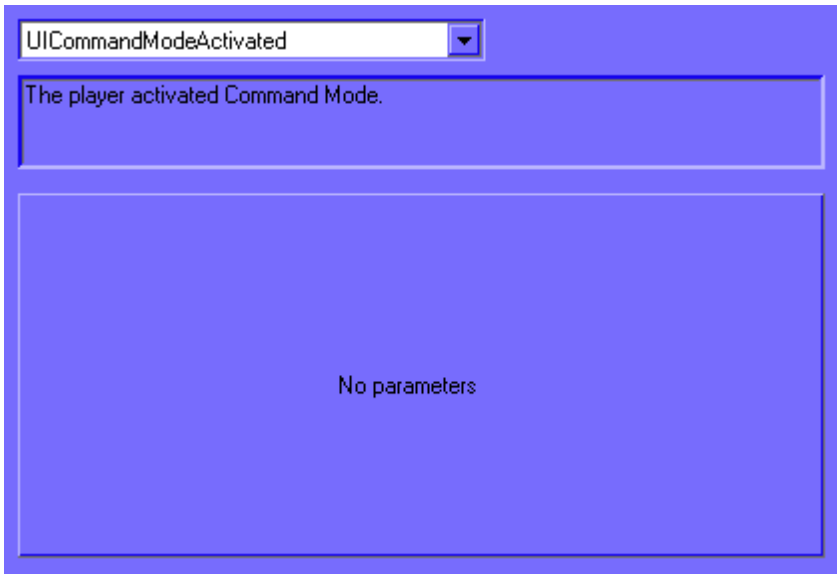
This Trigger activates when the designated Timer runs out of time. The only Parameter is Timer,

TimerExpiredforGame



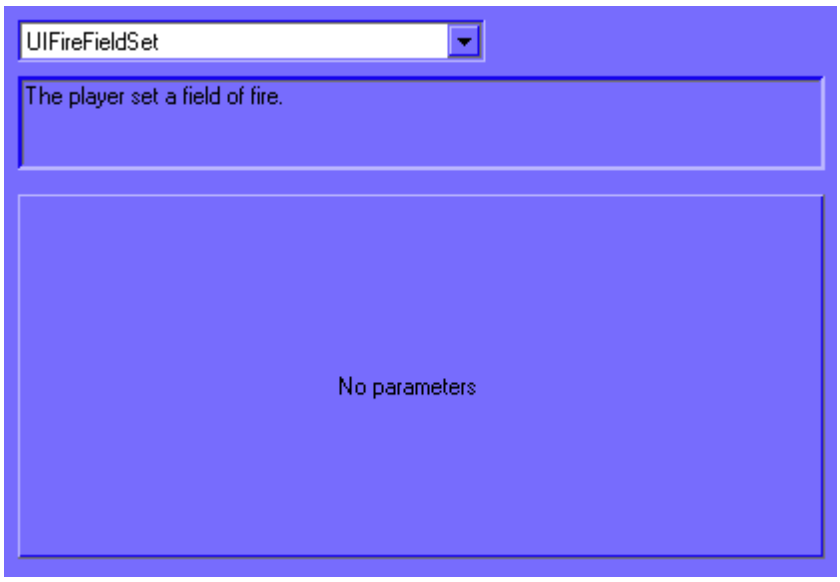
This Trigger activates when the Game Timer runs out. There are no Parameters for this Trigger, and it is used only in Multiplayer. See “In-Depth Multiplayer Scripting” for more information.

UICommandModeActivated



This Trigger activates when the player brings up the Command Map during the action phase of the game. It has no Parameters. Note that it is only for use in Single-Player missions, and not for any multiplayer modes including co-op.

UIFireFieldSet



This Trigger activates when the player designates a cover arc using the Command Map during the action phase. It has no Parameters and is intended only for single-player missions.

UIROESet

This Trigger activates when the player sets the ROE in the action phase of the game. It has no Parameters and is intended only for single-player missions.

UITeamSelected

This Trigger Activates when the player selects a Team using the Command Map in the action phase of the game. It has no Parameters and is intended only for single-player missions.

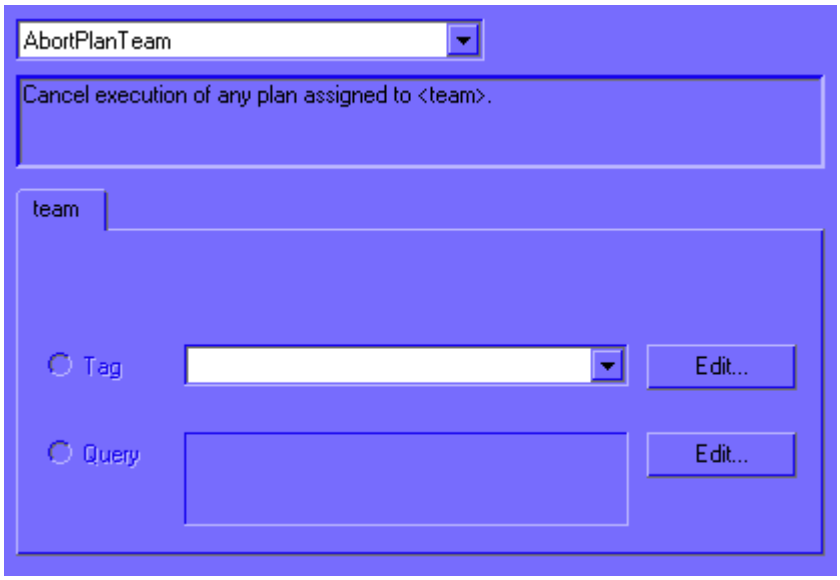
UIWaypointSet

This Trigger activates when the player sets a waypoint using the Command Map in the action phase of the game. It has no Parameters and is intended only for single-player missions.

Responses

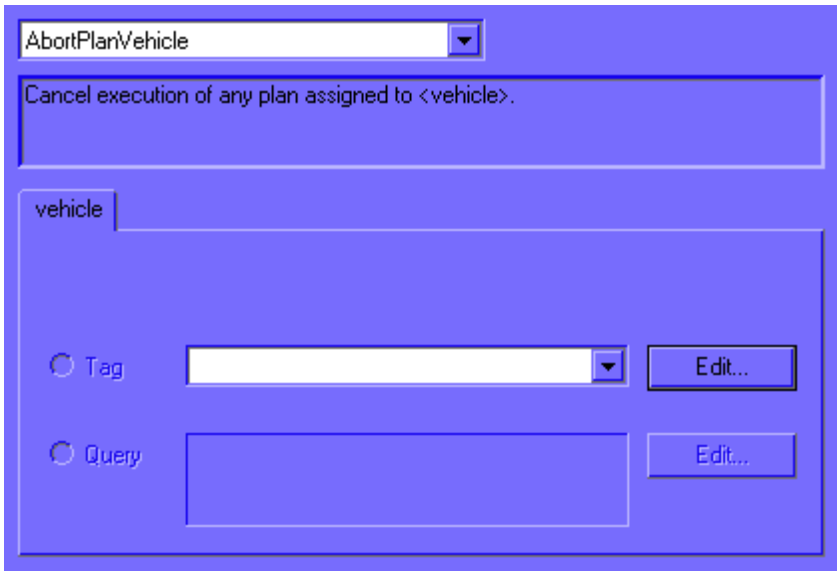
This section covers all the Responses in Igor, provides a description of their function, and examples of their use. Note that most Responses have Parameters that can be Queried. See the "Queries" section for more information.

AbortPlanTeam



This Response makes a Team cease whatever Plan they are currently executing. The only Parameter is Team,

AbortPlanVehicle



This Response makes a Vehicle cease whatever Plan it is currently executing. The only Parameter is Vehicle.

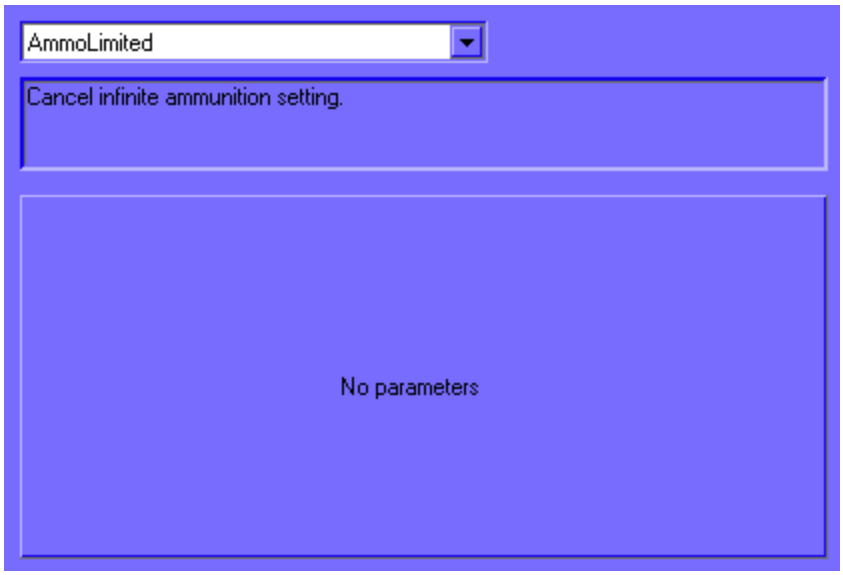
ActorStatsBoost

This Response was never implemented. If it were, it would allow the user to increase the statistics of the designated Actor.

ActorStatsRestore

This Response was never implemented. If it were, it would allow the user to designate an Actor that would have its stats returned to their value in its Actor File.

AmmoLimited

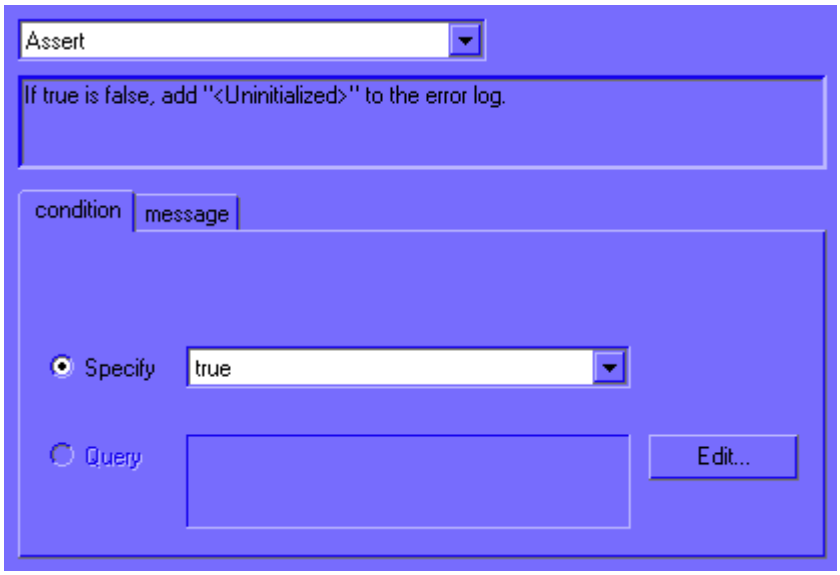


This Response cancels the AmmoUnlimited Response. It has no Parameters

AmmoUnlimited

This Response gives all Actors unlimited ammunition for weapons that use bullets. It has no Parameters.

Assert



This Response sends a user-created message to the Debug.txt file when running the debug version of Ghost Recon when the designated conditions are true. The Parameters are Condition and Message. Note that this Response is only intended for RSE use as the debug version of Ghost Recon is not for public release.

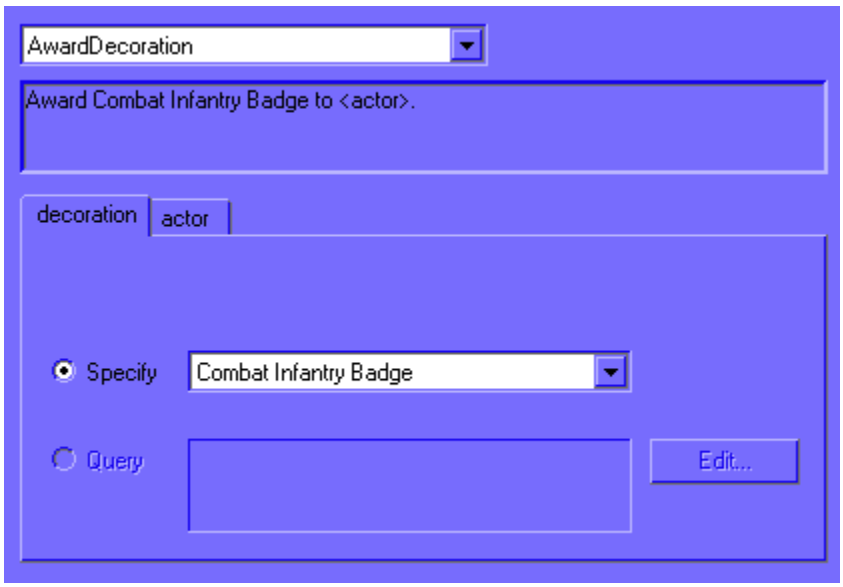
AvatarSwitchingOff

This Response prevents players from switching from character to character during the action phase of GR. It has no Parameters.

AvatarSwitchingOn

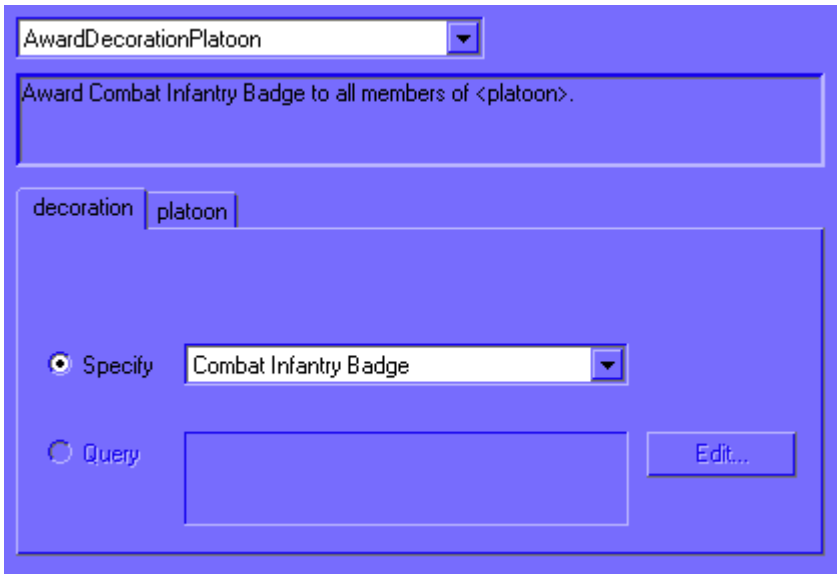
This Response allows players to switch from character to character during the action phase of GR. Avatar Switching is on by default. It has no Parameters.

AwardDecoration



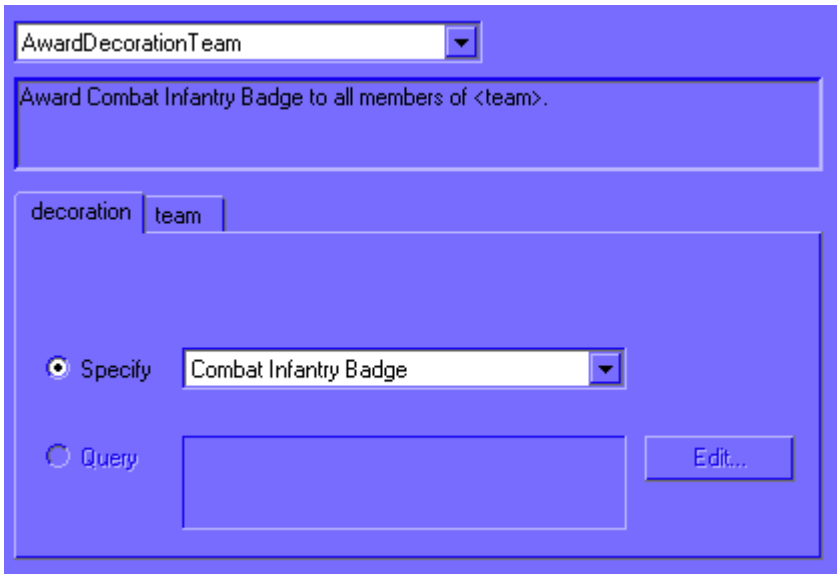
This Response awards the designated Actor with the selected Decoration. It has two Parameters, Decoration and Actor. Note that some decorations, such as ones rewarded for accuracy, number of kills, being wounded, etc, do not need to be awarded through this Response. The only Decorations that need to be awarded through this Response are campaign ribbons.

AwardDecorationPlatoon



This Response awards the designated Platoon with the selected Decoration. It has two Parameters, Decoration and Platoon. Note that some decorations, such as ones rewarded for accuracy, number of kills, being wounded, etc, do not need to be awarded through this Response. The only Decorations that need to be awarded through this Response are campaign ribbons.

AwardDecorationTeam

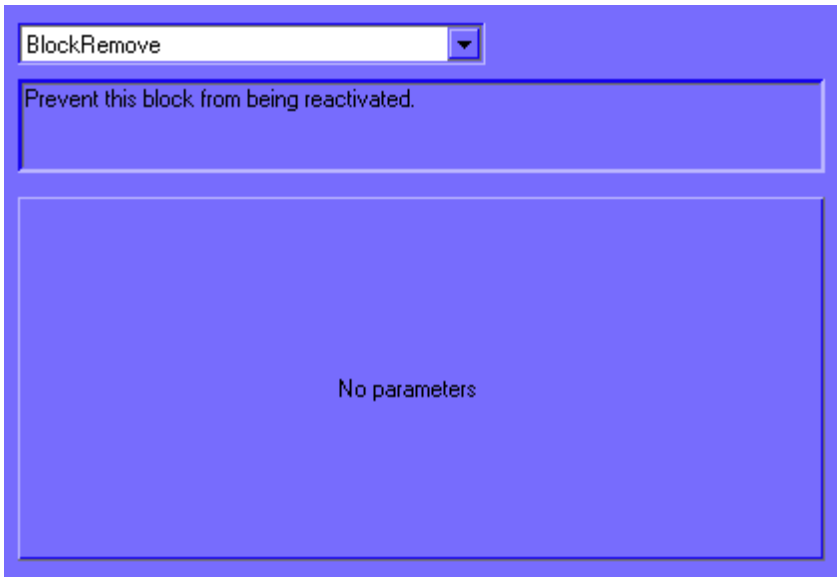


This Response awards the designated Team with the selected Decoration. It has two Parameters, Decoration and Team. Note that some decorations, such as ones rewarded for accuracy, number of kills, being wounded, etc, do not need to be awarded through this Response. The only Decorations that need to be awarded through this Response are campaign ribbons.

BlockPreserve

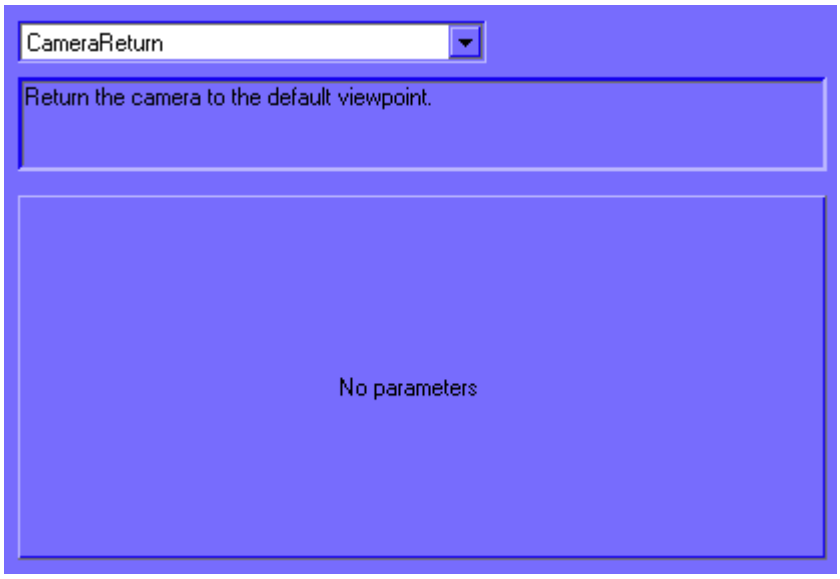
This Response enables a block to be reactivated every time its Trigger Event occurs. This Response has no Parameters.

BlockRemove



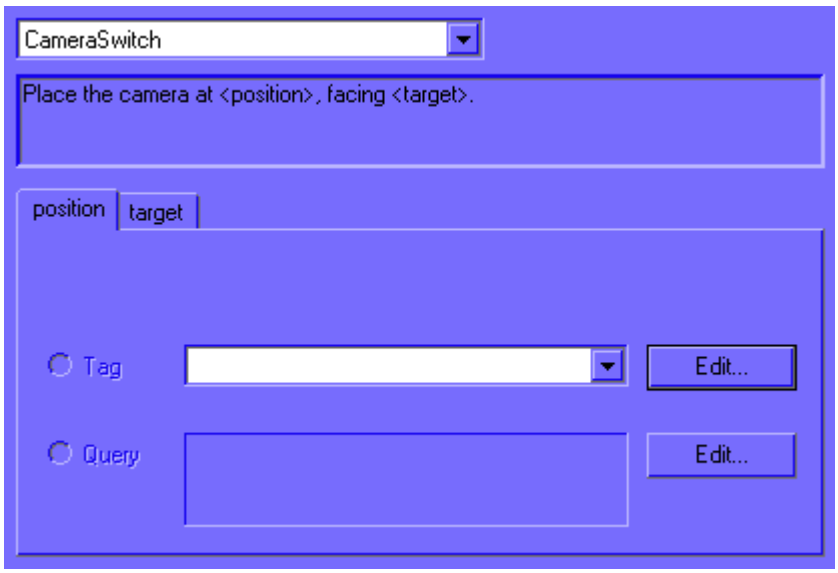
This Response prevents a block from being reactivated. By default, blocks will not be reactivated. This Response has no Parameters.

CameraReturn



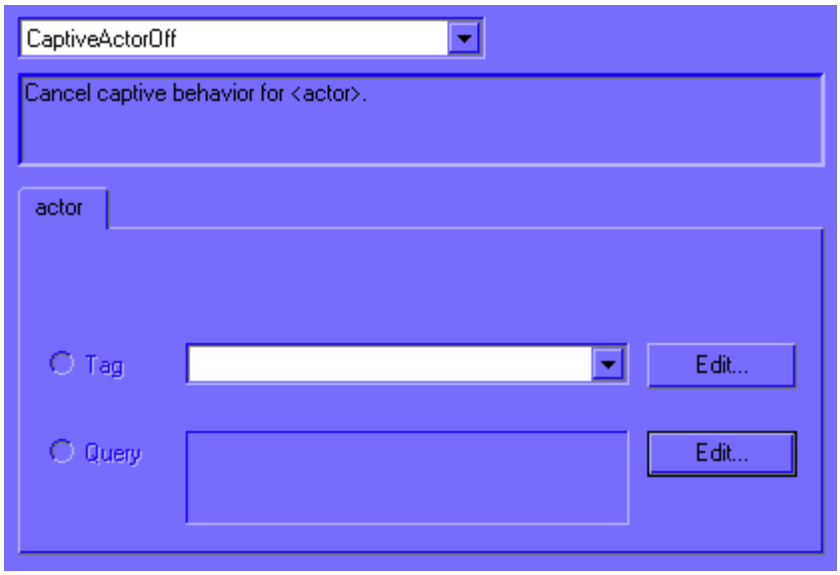
This Response returns the camera view to its default location – the player's view. It has no Parameters.

CameraSwitch



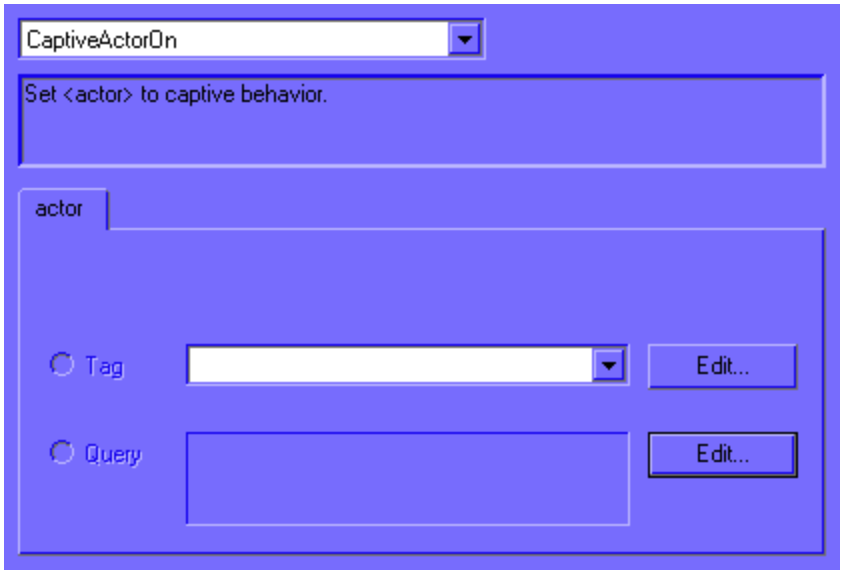
This Response forces the view to change to the selected Element, pointing at another Element. The Parameters are Position and Target.

CaptiveActorOff



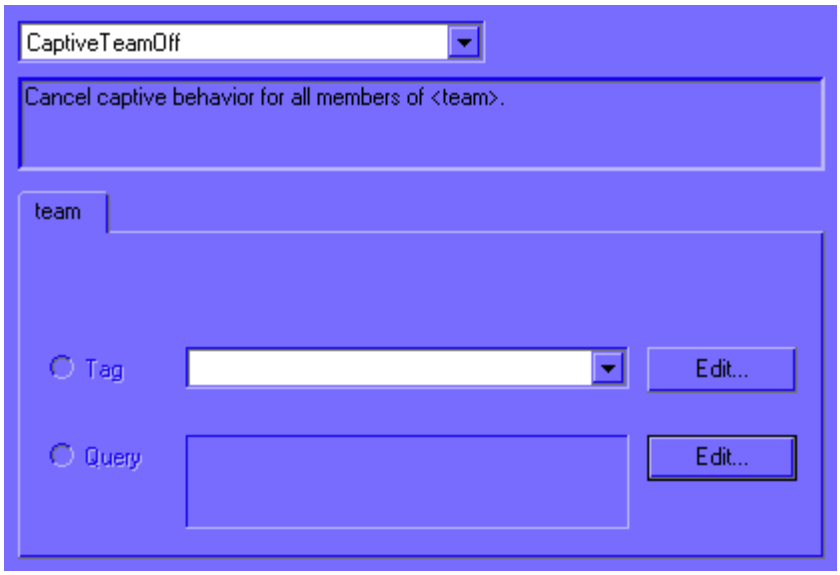
This Response cancels the captive behavior for the designated Actor. Its only Parameter is Actor.

CaptiveActorOn



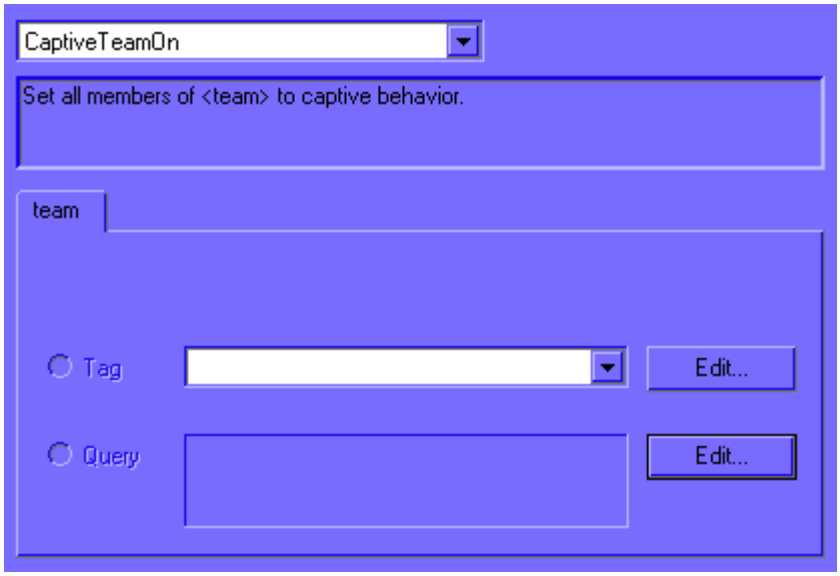
This Response activates the captive behavior for the designated Actor. The only Parameter is Actor.

CaptiveTeamOff



This Response cancels the captive behavior for the designated Team. Its only Parameter is Team.

CaptiveTeamOn

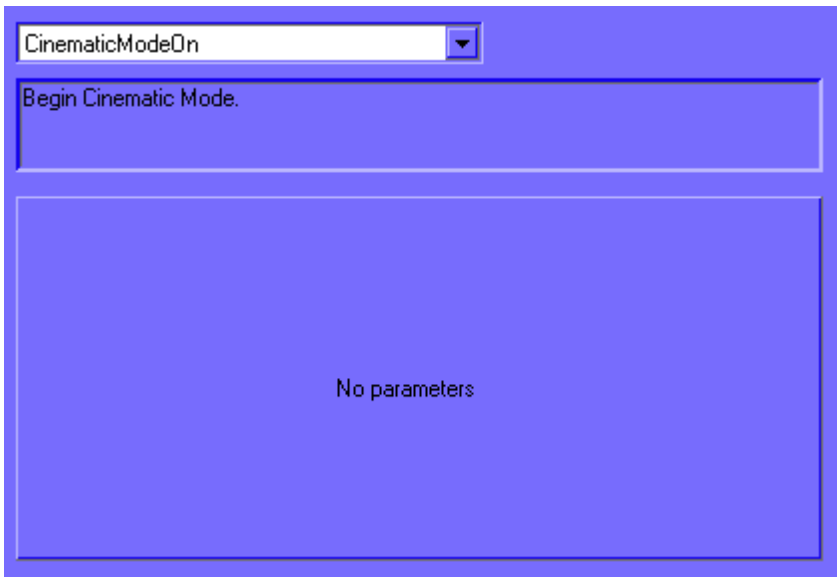


This Response activates the captive behavior for the designated Team. Its only Parameter is Team.

CinematicModeOff

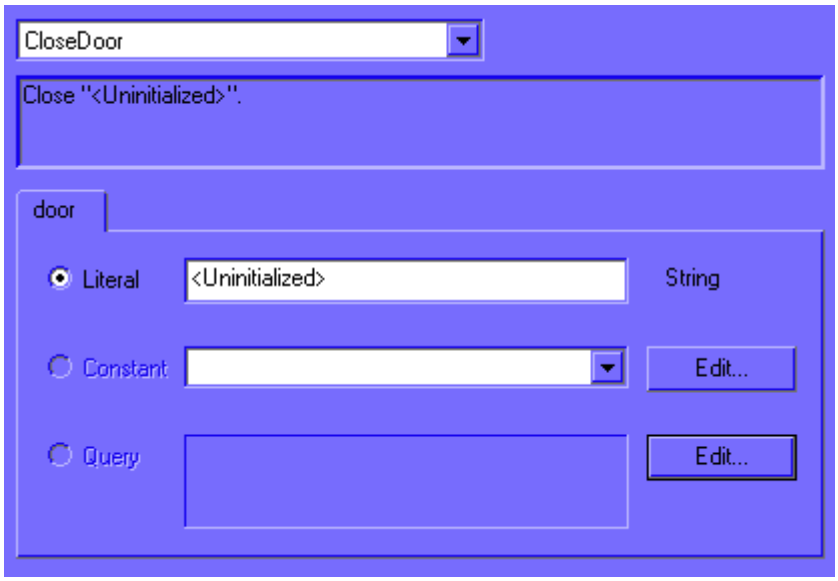
This Response cancels the cinematic mode. It has no Parameters.

CinematicModeOn



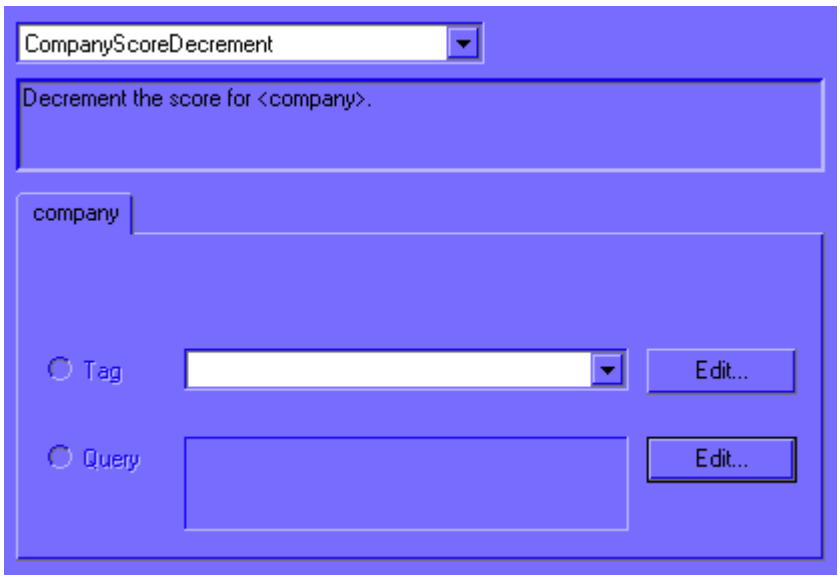
This Response begins the cinematic mode, which disables user control of the game and puts letterboxes on the top and bottom of the screen. It has no Parameters.

CloseDoor



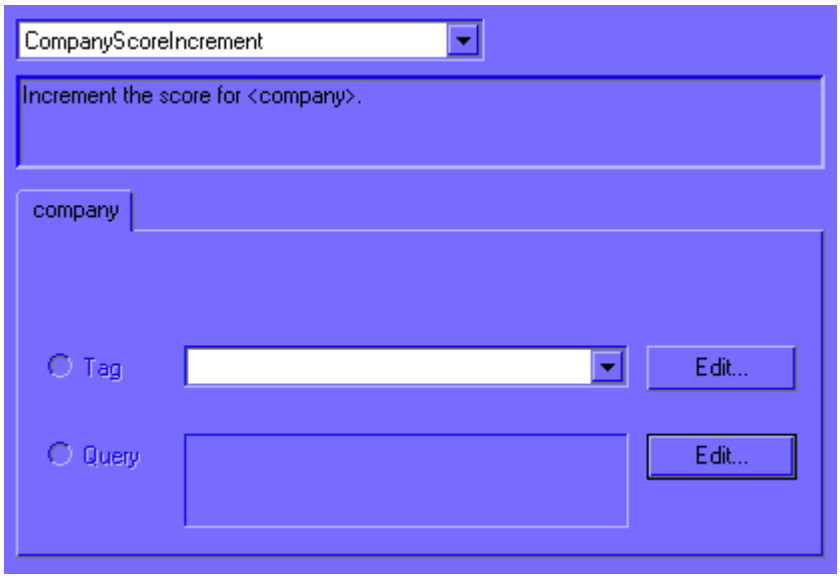
This Response makes the designated Door close. If the Door is already closed, nothing will happen. All Doors begin closed by default. The only Parameter is Door, which can be entered exactly as it appears when viewing Map Objects in Igor, from a pull-down menu as a constant, or Queried.

CompanyScoreDecrement



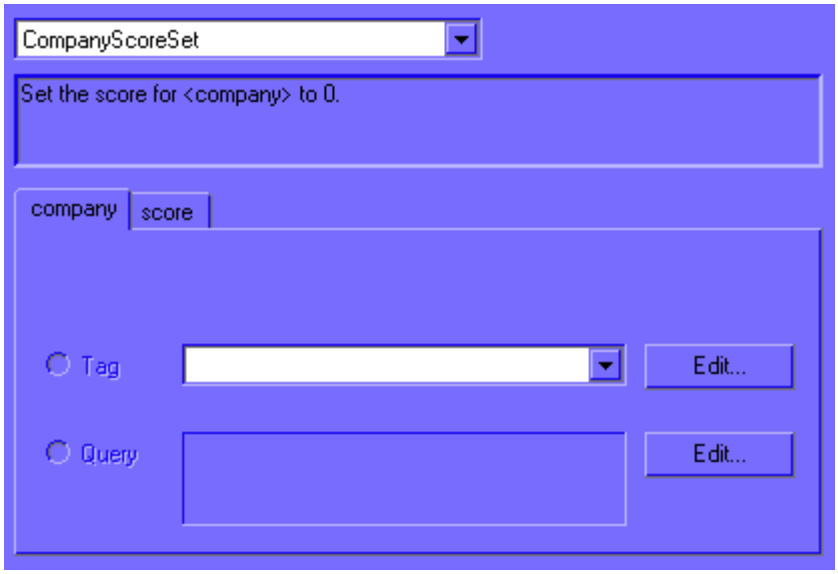
This Response decreases the score for the designated Company by 1. Its only Parameter is Company, which can be selected from a pull-down menu or Queried. Scores are usually only found in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

CompanyScoreIncrement



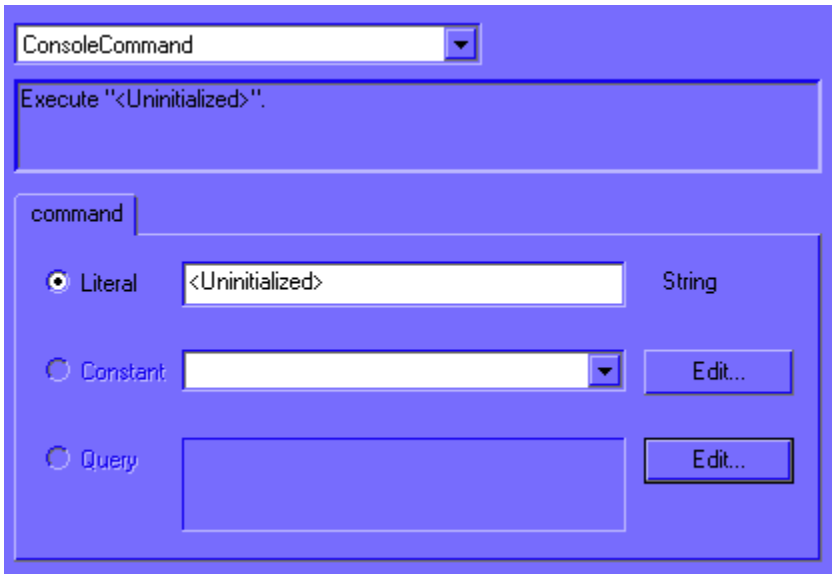
This Response increases the score for the designated Company by 1. Its only Parameter is Company, which can be selected from a pull-down menu or Queried. Scores are usually only found in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

CompanyScoreSet



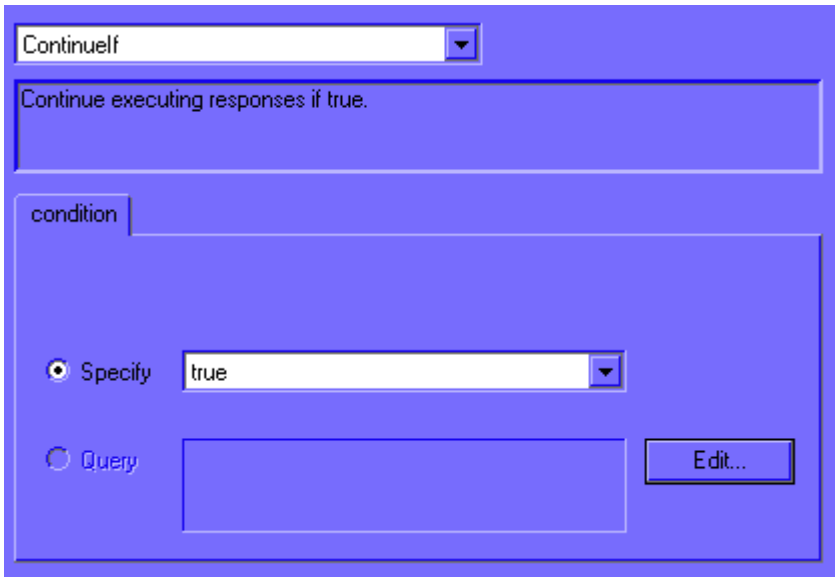
This Response sets the score for the designated company to a user-defined value. The Parameters are Company and Score. Company can be selected from a pull-down menu or Queried, while Score can be entered as a literal, selected from a pull-down menu as a constant, or Queried. Scores are usually only found in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

ConlsoleCommand



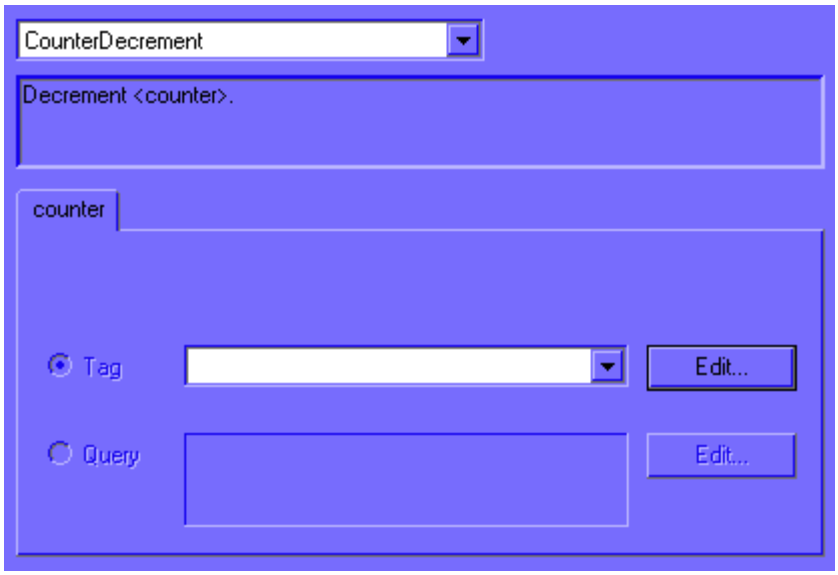
This Response activates the designated console command as if the user had brought up the console in-game and typed in a console command. The only Parameter is Command, which can be entered as a literal, selected from a pull-down menu as a constant, or Queried. Note that this Response should not be used to activate or deactivate any Actor state (such as invisibility or invincibility) if there are scripting Responses that do the same thing due to the user having the ability to type in these very same console commands during the game.

ContinueIf



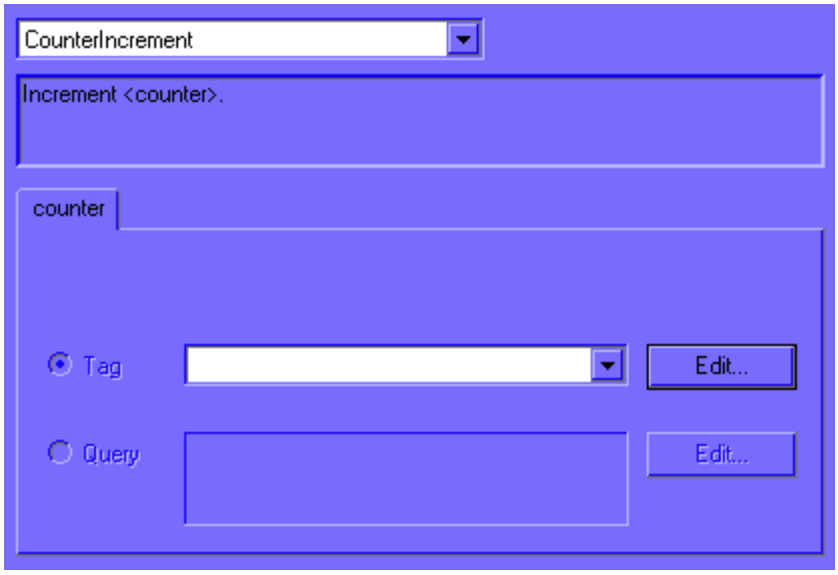
This Response determines if the Responses below it will be activated or not depending on certain conditions. If the conditions are true, the other Responses will execute. It's only Parameter is Condition.

CounterDecrement



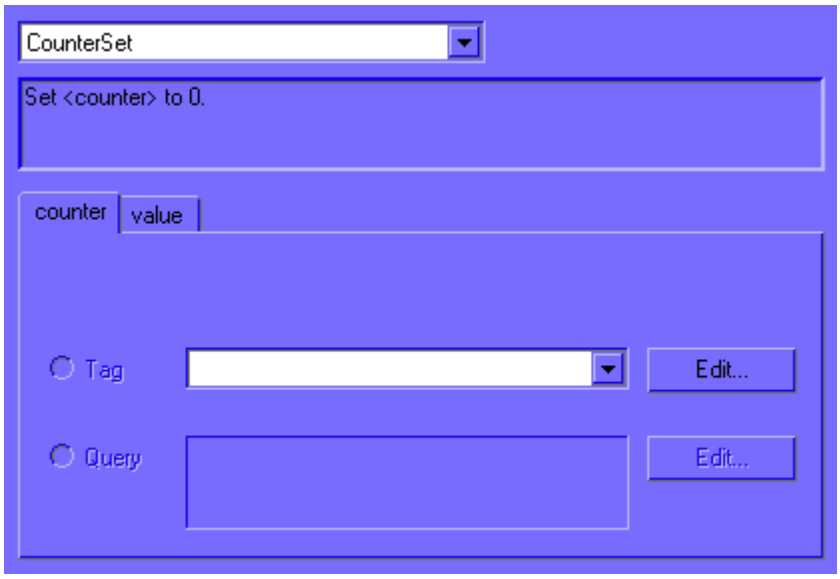
This Response lowers the value of the designated Counter by 1. Its only Parameter is Counter,

CounterIncrement



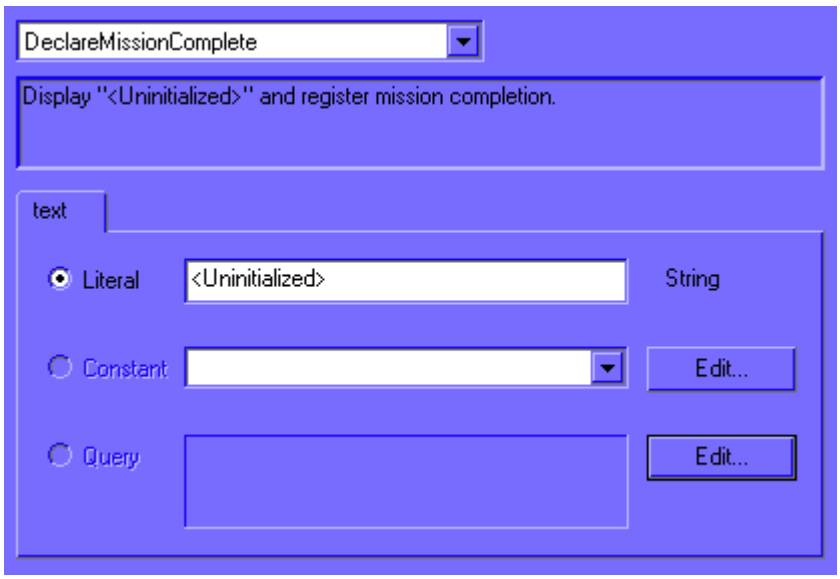
This Response raises the value of the designated Counter by 1. Its only Parameter is Counter,

CounterSet



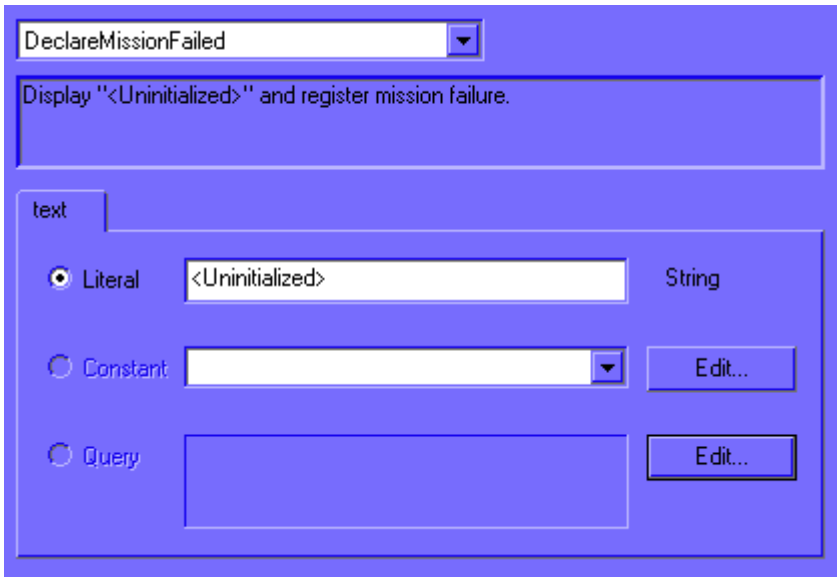
This Response sets a Counter to a user-defined value. Its Parameters are Counter and Value. The Counter is selected from a pull-down menu, while Value can be entered as a literal, selected from a pull-down menu as a constant, or Queried.

DeclareMissionComplete



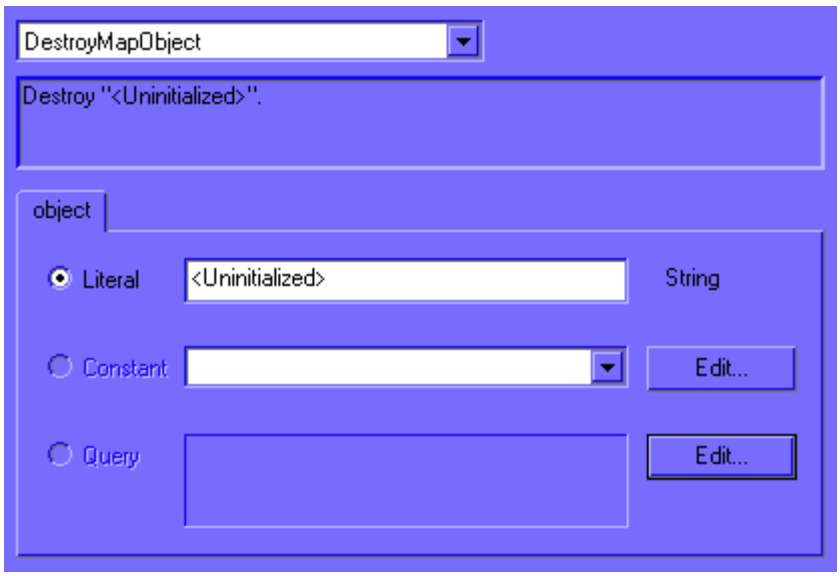
This Response displays a message and ends the mission as a success. The only Parameter is Text, which can be entered as a Literal, selected from a pull-down menu as a constant, or Queried.

DeclareMissionFailed



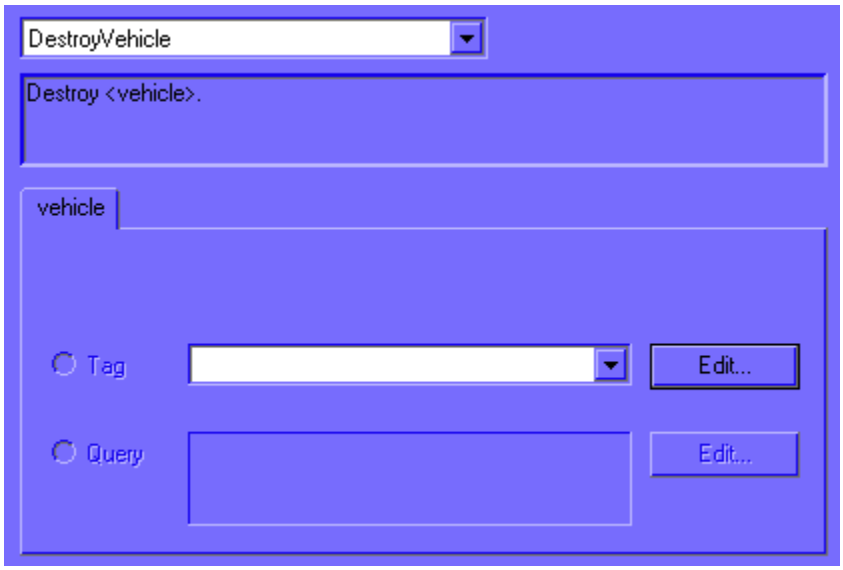
This Response displays a message and ends the mission as a failure. The only Parameter is Text, which can be entered as a literal, selected from a pull-down menu as a constant, or Queried.

DestroyMapObject



This Response destroys the designated Map Object. The only Parameter is Object, which can be entered exactly as it appears when Map Objects are viewed in Igor as a literal, selected from a pull-down menu as a constant, or Queried.

DestroyVehicle

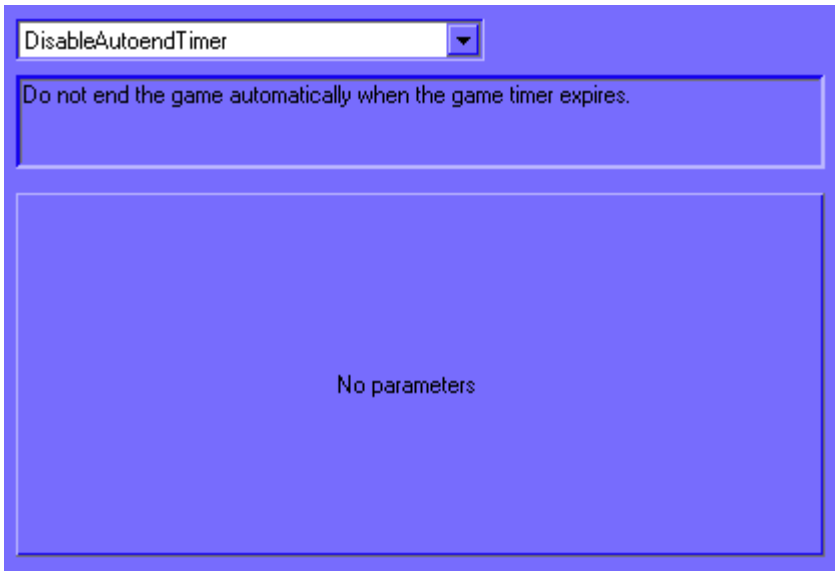


This Response destroys the designated Vehicle. Its only Parameter is Vehicle, Note that this Response will NOT cause the DeathVehicle Trigger Event to activate.

DisableAutoendNoPlayers

This Response forces the game to continue even when no players are left playing. It has no Parameters. This Response is mainly used in multiplayer game types. For more information refer to the In-Depth Multiplayer Scripting section.

DisableAutoendTimer



This Response forces the game to continue after the game timer expires. It has no Parameters. This Response is mainly used in multiplayer game types. For more information refer to the In-Depth Multiplayer Scripting section.

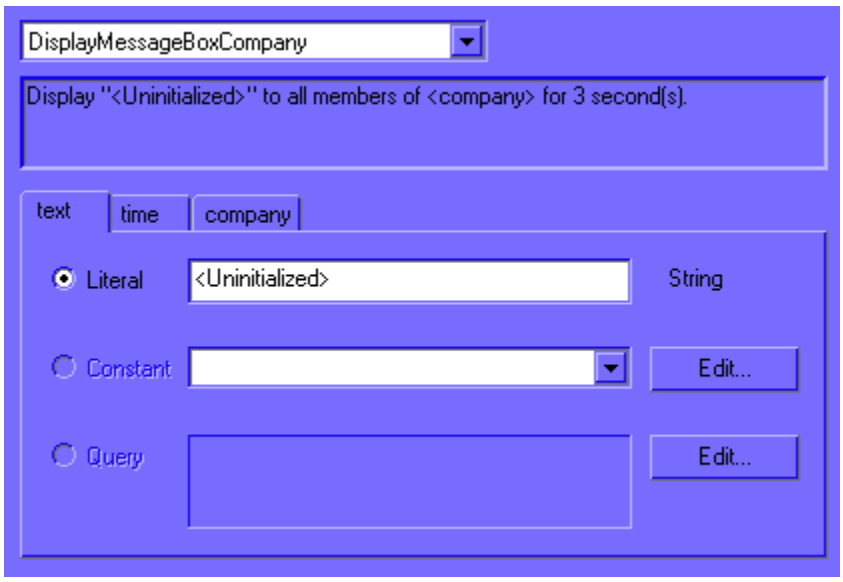
DisplayMessageAll

This Response displays a user-defined message to all players in the game as if the message were typed into the chat dialog during the game. Its only Parameter is Text, which can be entered as a literal, selected from a pull-down menu as a constant, or Queried.

DisplayMessageBoxAll

This Response displays a user-defined message to all players in the game in the message box in the upper center of the screen for a specified amount of time. The Parameters are Text and Time.

DisplayMessageBoxCompany

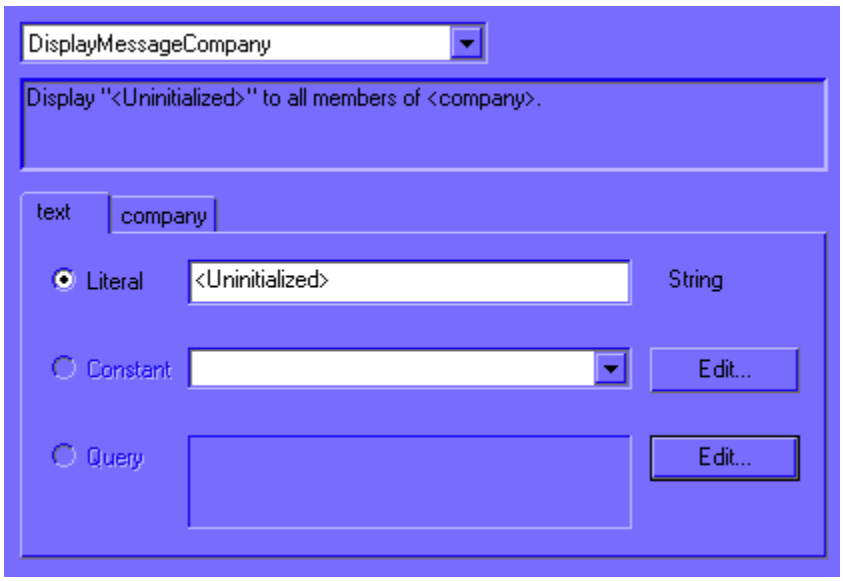


This Response displays a user-defined message to all players within a designated Company for a specified amount of time. The Parameters are Text, Time, and Company.

DisplayMessageBoxPlayer

This Response displays a user-defined message to the designated player for a specified amount of time. The Parameters are Text, Time, and Player.

DisplayMessageCompany



This Response displays a user-defined message to the designated company as if it were typed into the chat dialog. Its Parameters are Text and Company.

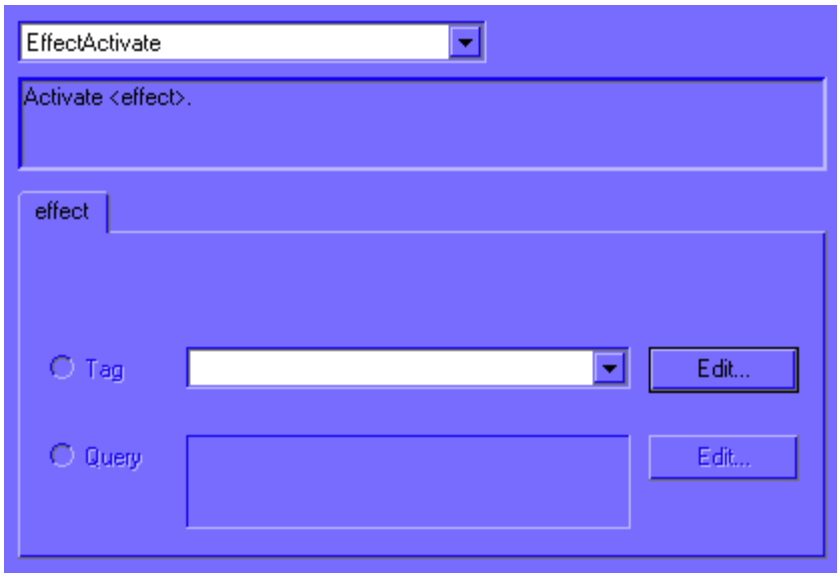
DisplayMessagePlayer

This Response displays a user-defined message to the designated player as if iter were typed into the chat dialog. Its Parameters are Text and Player.

DisplayVoiceover

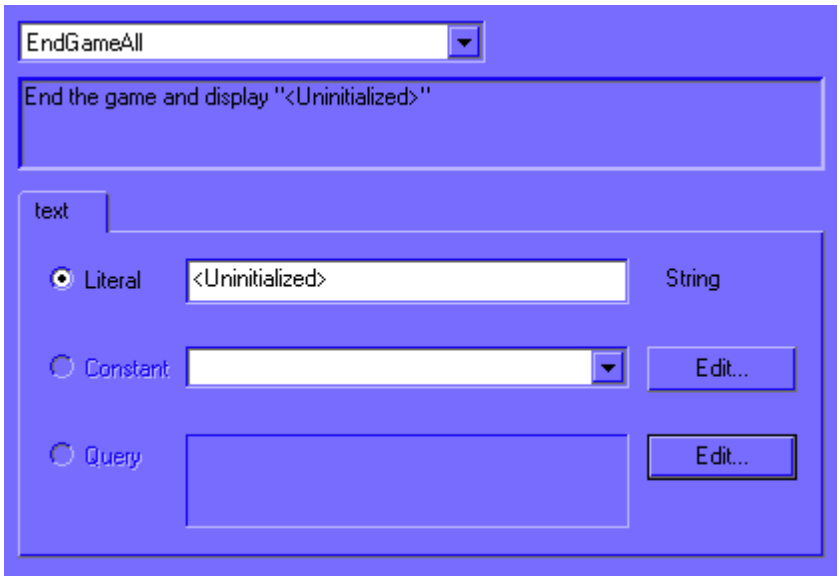
This Response displays the user-defined text in the message box while the specified wav file is playing at the designated volume. The message will vanish when the wav file is done playing. Its Parameters are Text, Sound, and Volume.

EffectActivate



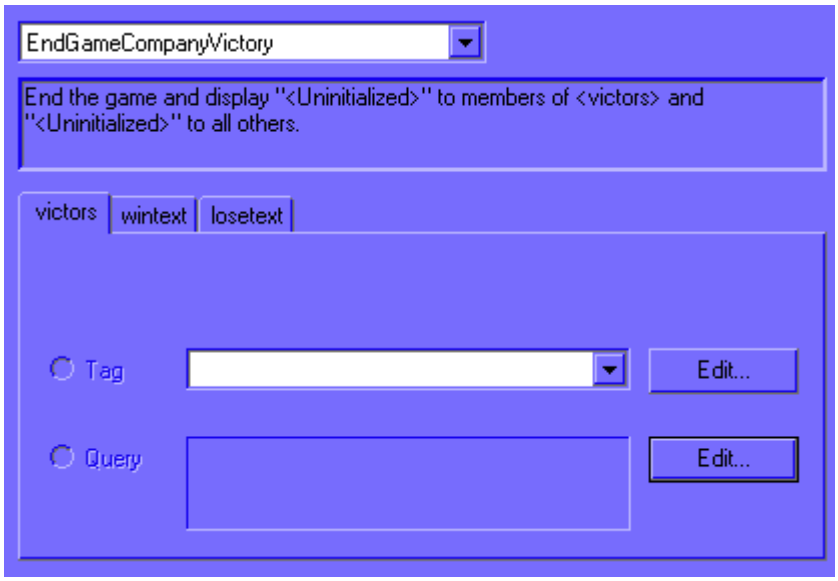
This Response activates the designated Effect. Its Parameter is Effect. For more information on Effects, see the Effects part of this section.

EndGameAll



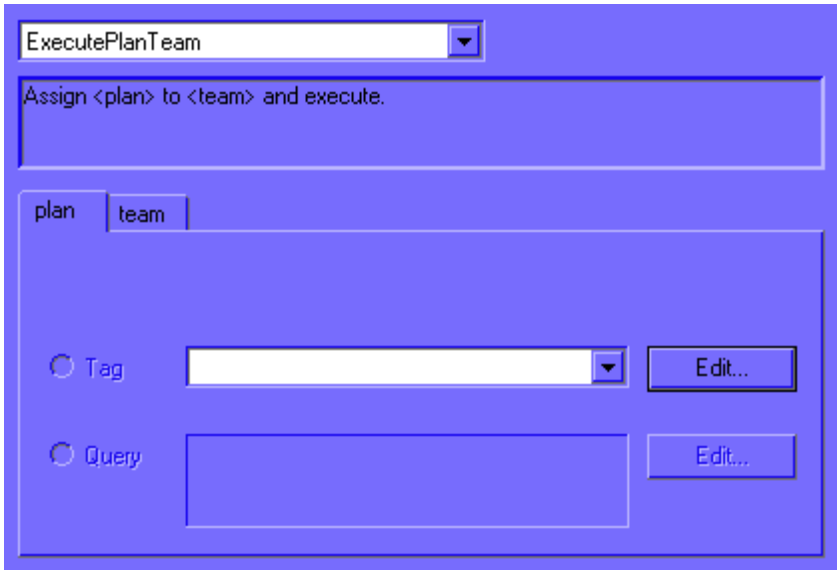
This Response ends the game for all players and displays the user-defined message in a message box as the game is ending. Its only Parameter is Text. This Response is mainly used in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

EndGameCompanyVictory



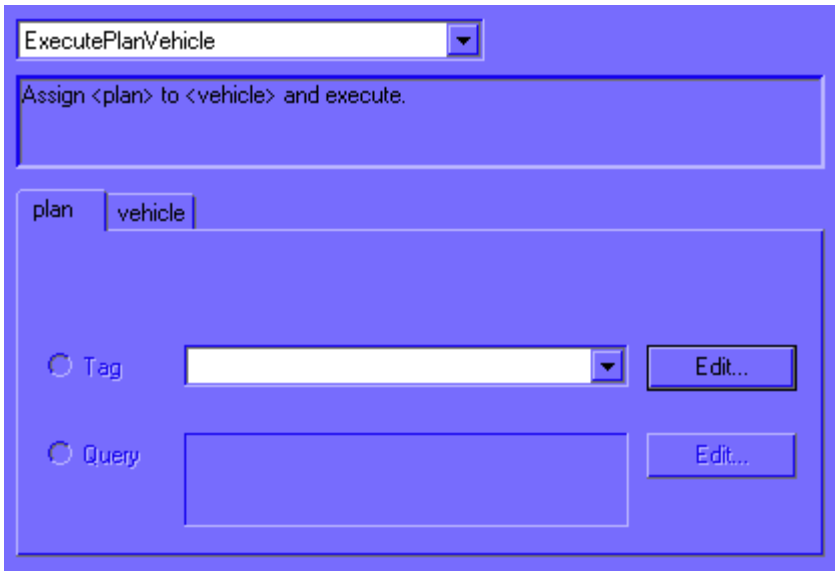
This Response ends the game and displays one user-defined message to the winners, and a different user-defined message to the losers. Its Parameters are Victors, Wintext, and Losetext. This Response is mainly used in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

ExecutePlanTeam



This Response assigns a Plan to a Team, who then executes that Plan. Its Parameters are Plan and Team.

ExecutePlanVehicle

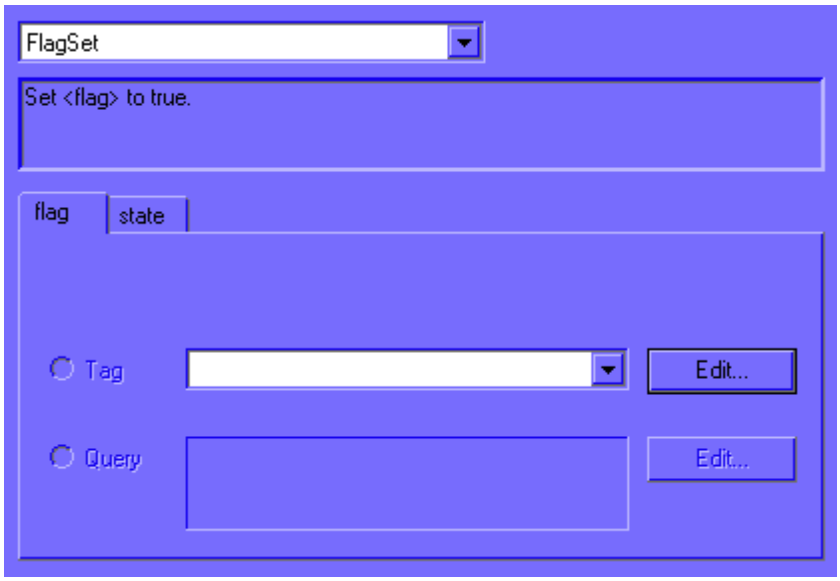


This Response assigns a Plan to a Vehicle, which then executes that Plan. Its Parameters are Plan and Team.

FadeOut

This Response forces the display to fade to black, and then from black back to normal. It has no Parameters.

FlagSet

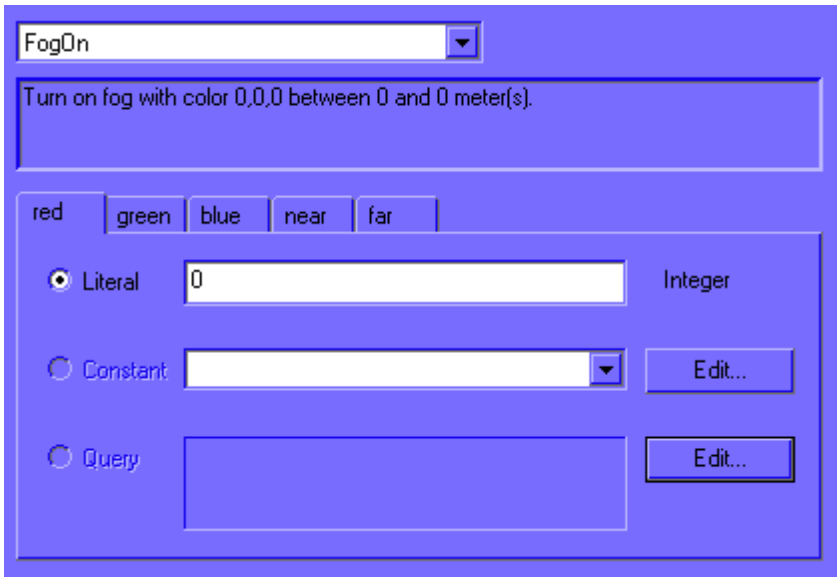


This Response changes the state of the designated flag to either true or false. The Parameters are Flag and State.

FogOff

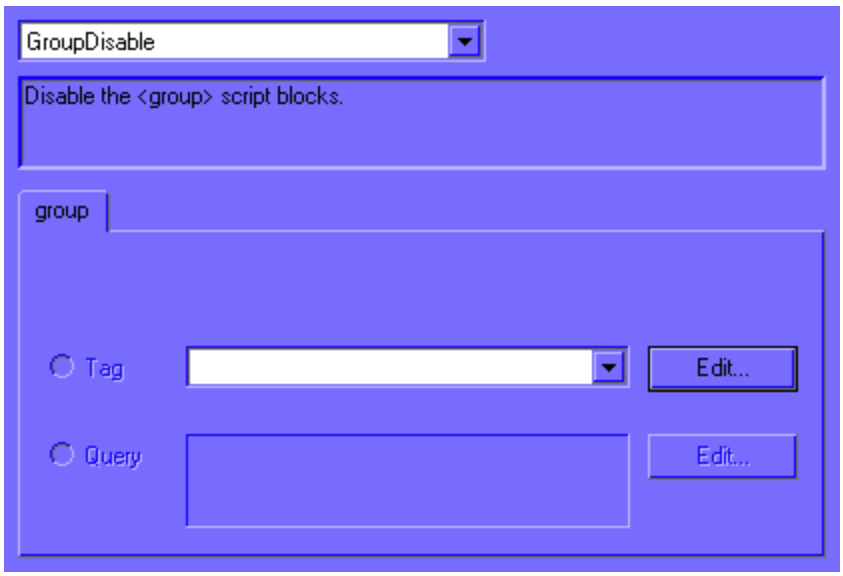
This Response turns off all fog effects in a mission, effectively overriding the environment file for the map. It has no Parameters.

FogOn



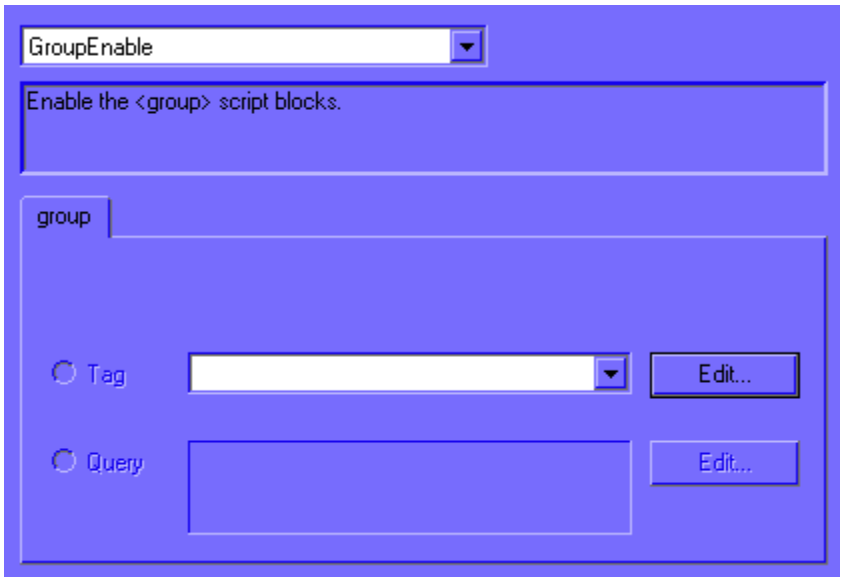
This Response turns fog on all fog effects in a mission, and lets the user define the properties of the fog. Its parameters are Red, Green, Blue, Near, and Far.

GroupDisable



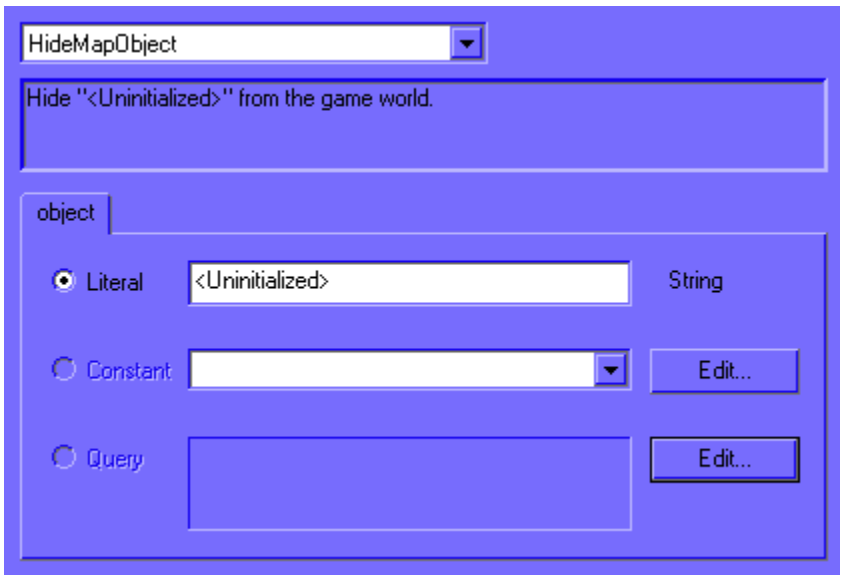
This Response disables all blocks within the designated Group. Its only Parameter is Group.

GroupEnable



This Response enables all blocks within the designated Group. Its only Parameter is Group.

HideMapObject

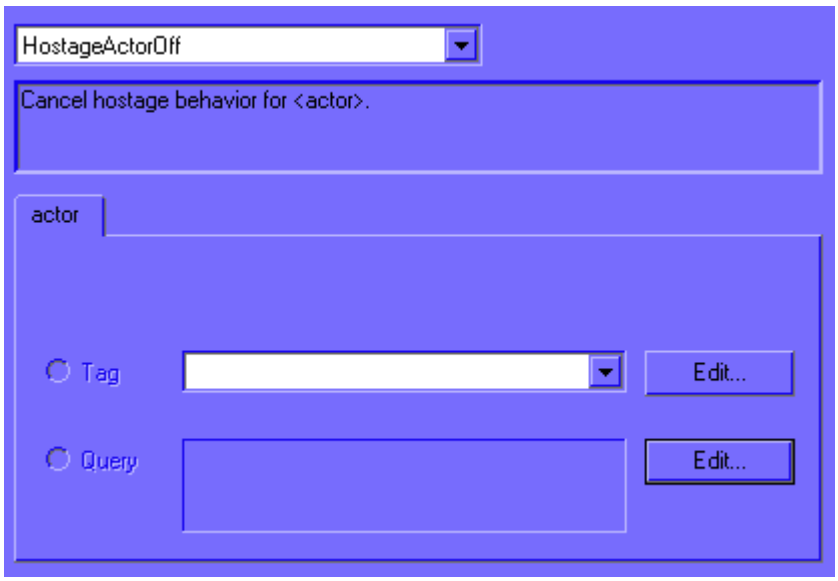


This Response makes the designated Map Object vanish from sight in the action phase of GR. Its Parameter is Object.

HideThing

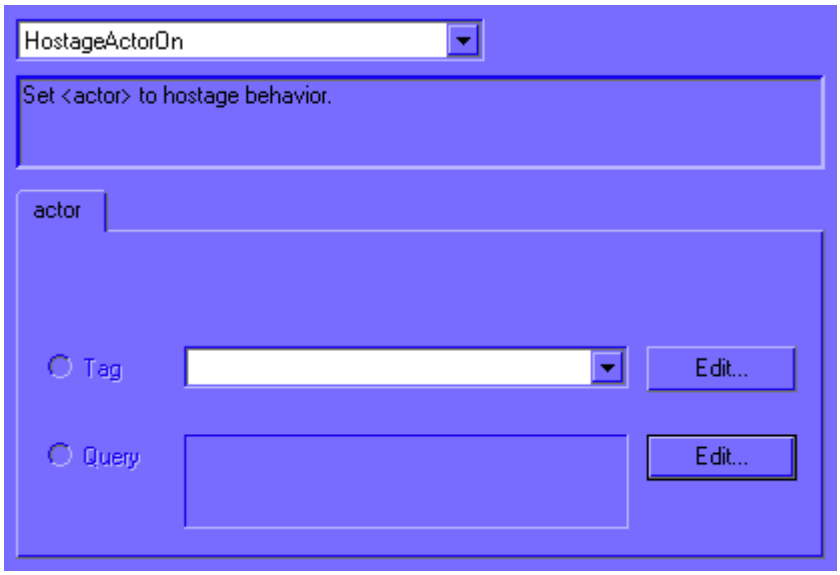
This Response makes the designate Element vanish from sight in the action phase of GR. Its Parameter is Thing.

HostageActorOff



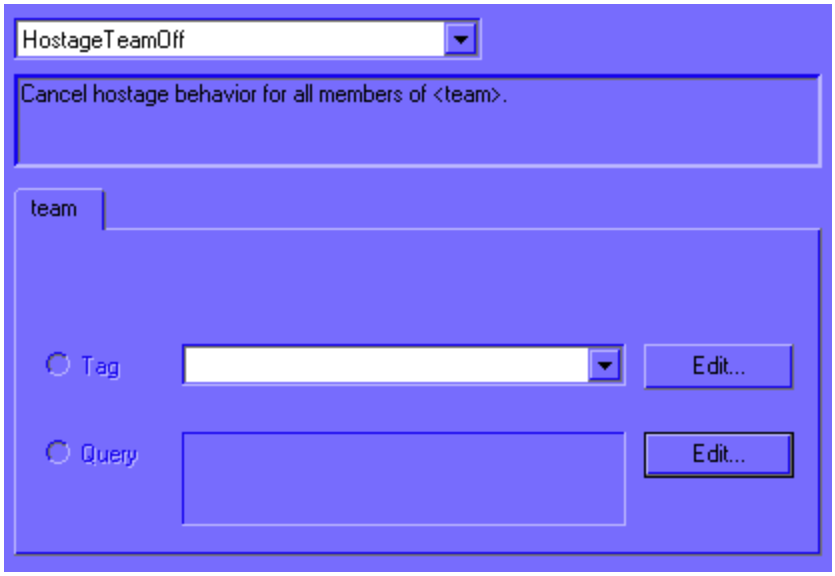
This Response makes the designated Actor cease the hostage behavior. Its only Parameter is Actor.

HostageActorOn



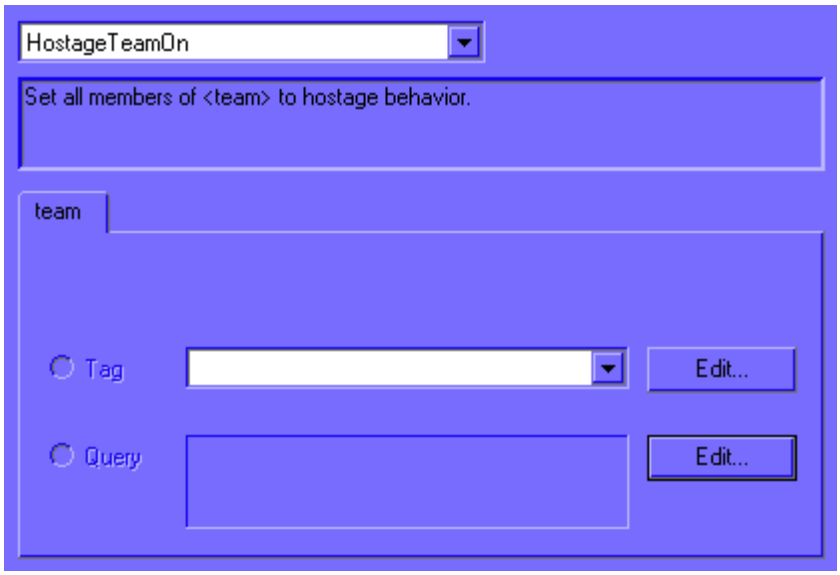
This Response makes the designated Actor begin the hostage behavior. Its only Parameter is Actor.

HostageTeamOff



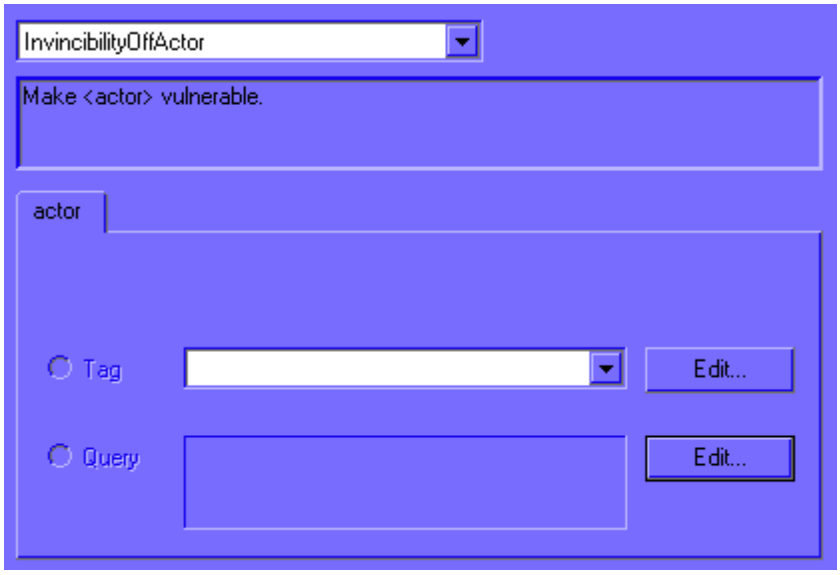
This Response makes all Actors in the designated Team cease the hostage behavior. Its only Parameter is Team.

HostageTeamOn



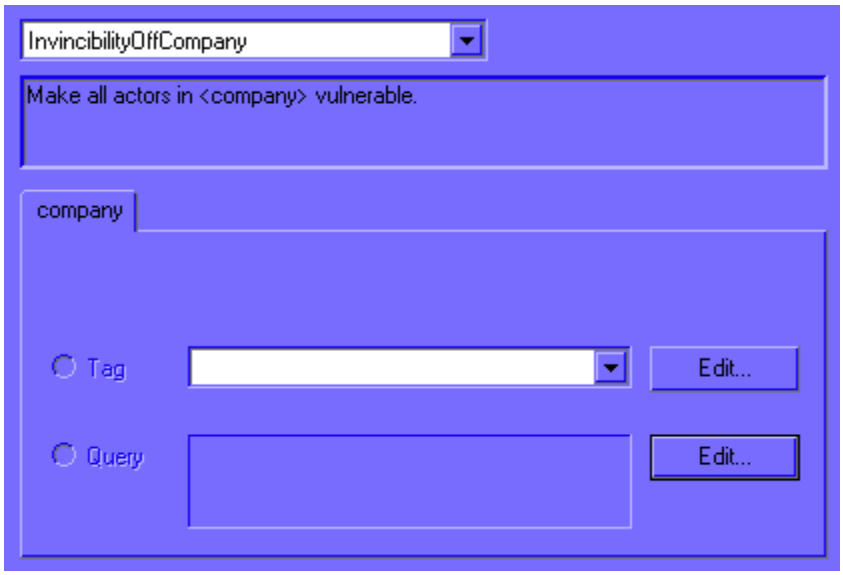
This Response makes all the Actors in the designated Team begin the hostage behavior. Its only Parameter is Team.

InvincibiltyOffActor



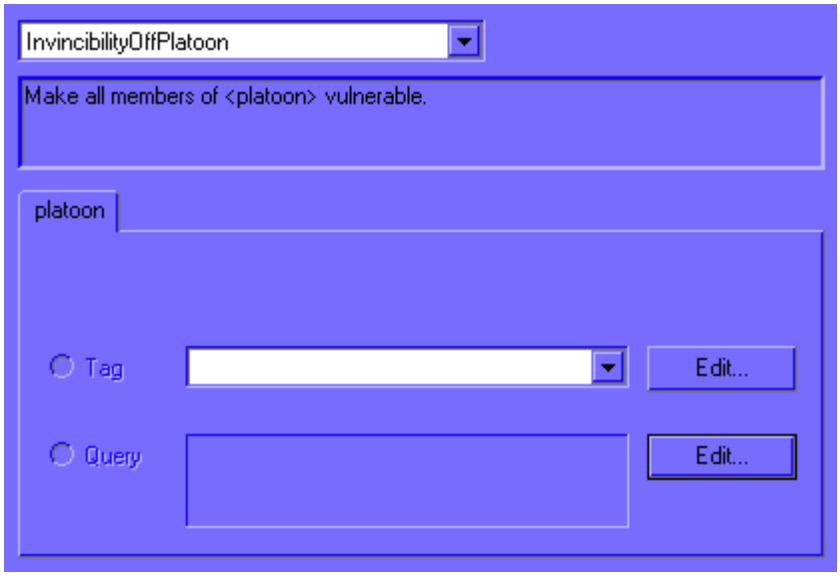
This Response turns invincibility off for the designated Actor. Its only Parameter is Actor.

InvincibiltyOffCompany



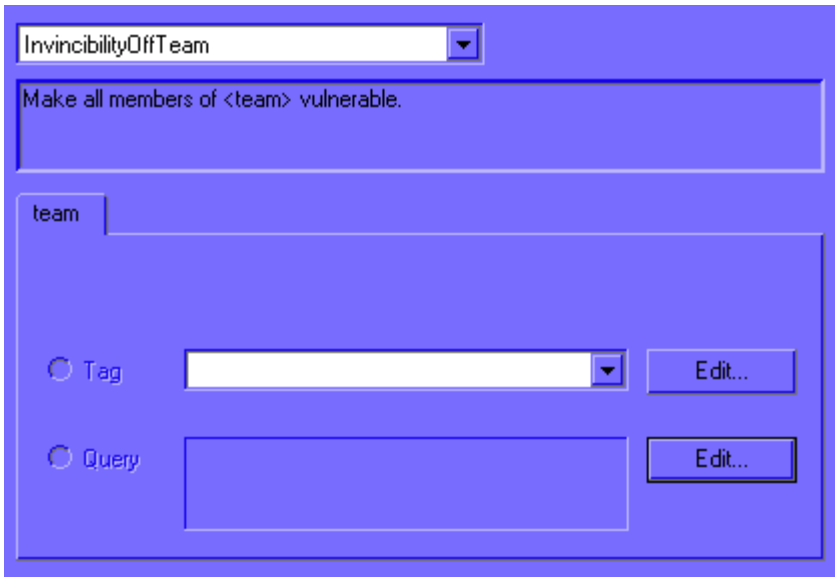
This Response turns invincibility off for all Actors in the designated Company. Its only Parameter is Company.

InvincibiltyOffPlatoon



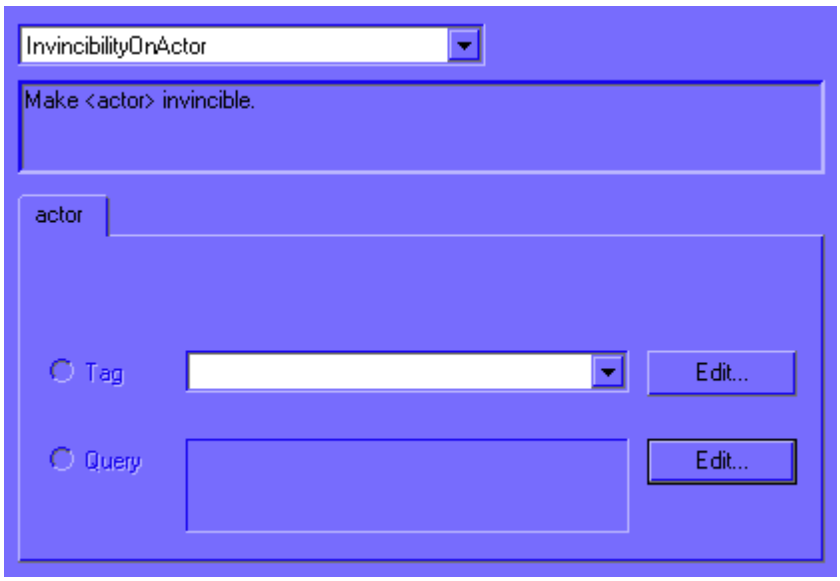
This Response turns invincibility off for the Actors in the designated Platoon. Its only Parameter is Platoon.

InvincibiltyOffTeam



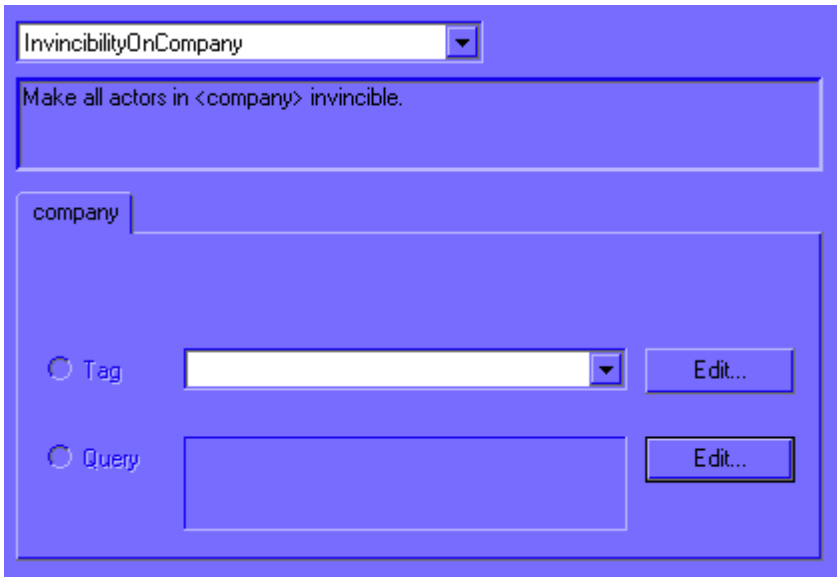
This Response turns invincibility off for all Actors in the designated Team. Its only Parameter is Team.

InvincibiltyOnActor



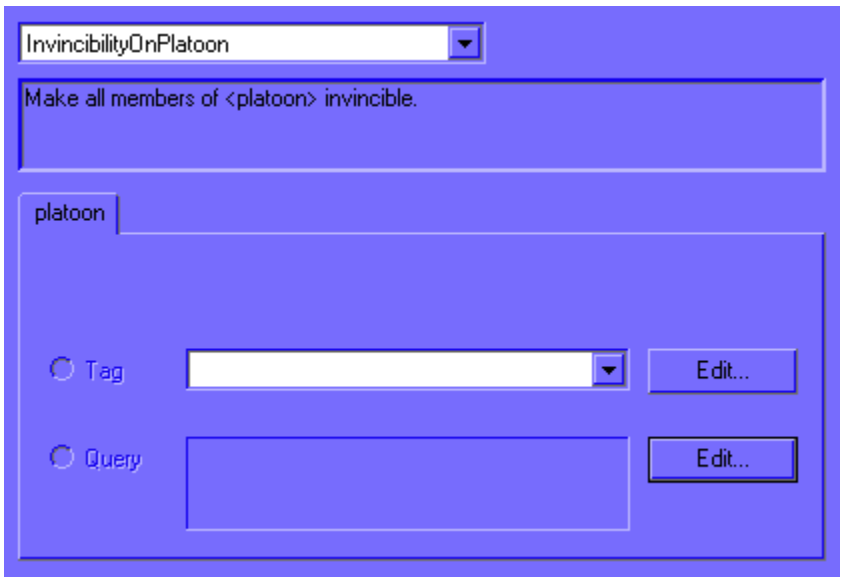
This Response turns invincibility on for the designated Actor. Its only Parameter is Actor.

InvincibiltyOnCompany



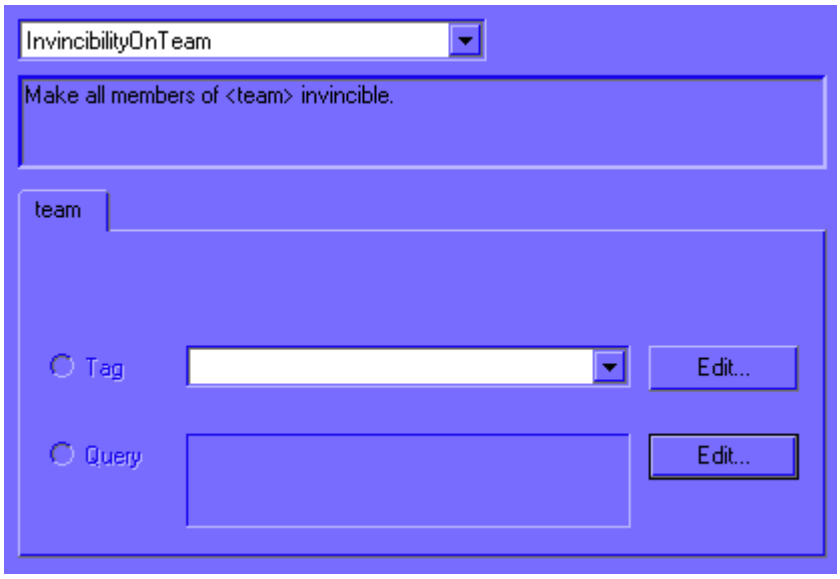
This Response turns invincibility on for all Actors in the designated Company. Its only Parameter is Company.

InvincibiltyOnPlatoon



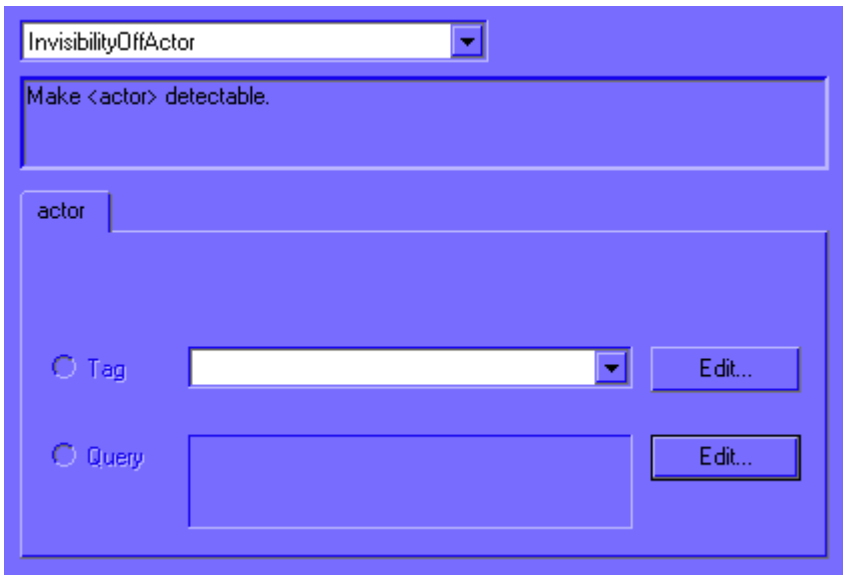
This Response turns invincibility on for all Actors in the designated Platoon. Its only Parameter is Platoon.

InvincibiltyOnTeam



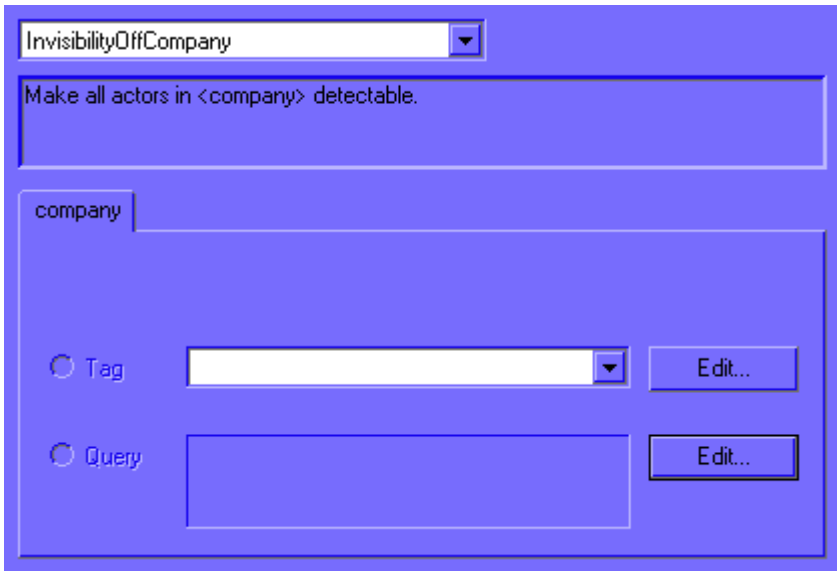
This Response turns invincibility on for all Actors in the designated Team. Its only Parameter is Team.

InvisibiltyOffActor



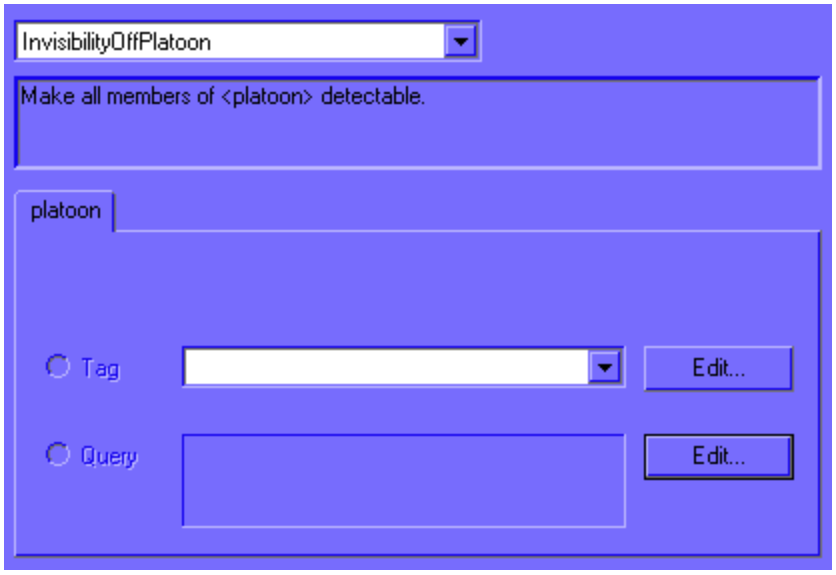
This Response turns invisibility off for the designated Actor. Its only Parameter is Actor. Note that invisibility doesn't make the Actor vanish from sight – it just makes other Actors unable to detect it.

InvisibiltyOffCompany



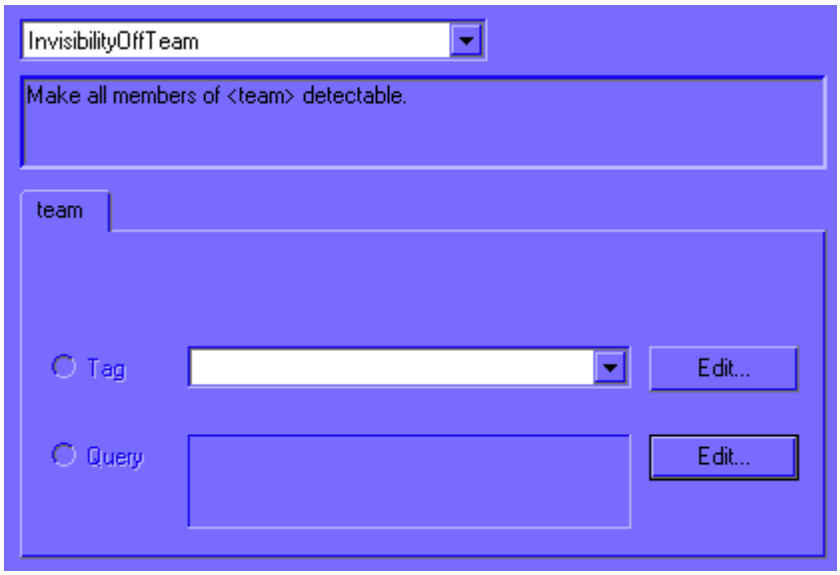
This Response turns invisibility off for Actors in the designated Company. Its only Parameter is Company. Note that invisibility doesn't make the Actors vanish from sight – it just makes other Actors unable to detect them.

InvisibiltyOffPlatoon



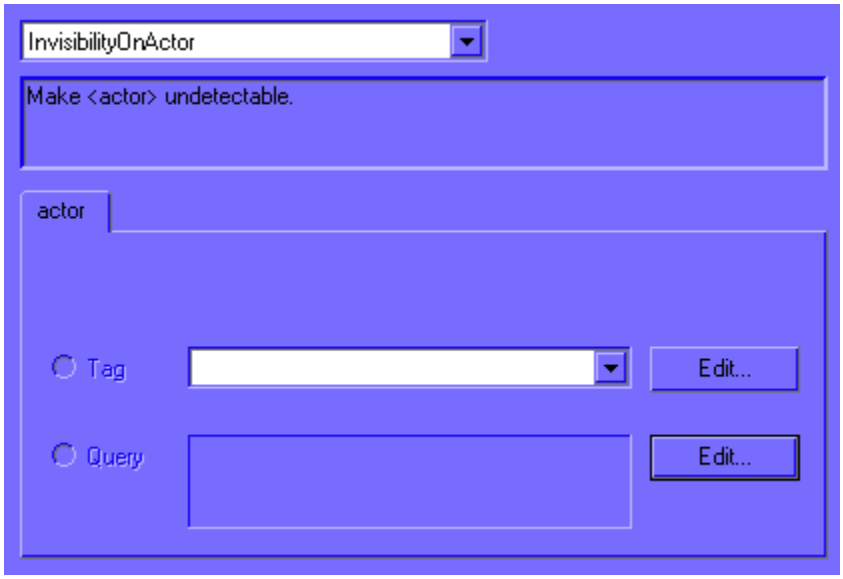
This Response turns invisibility off for Actors in the designated Platoon. Its only Parameter is Platoon. Note that invisibility doesn't make the Actors vanish from sight – it just makes other Actors unable to detect them.

InvisibiltyOffTeam



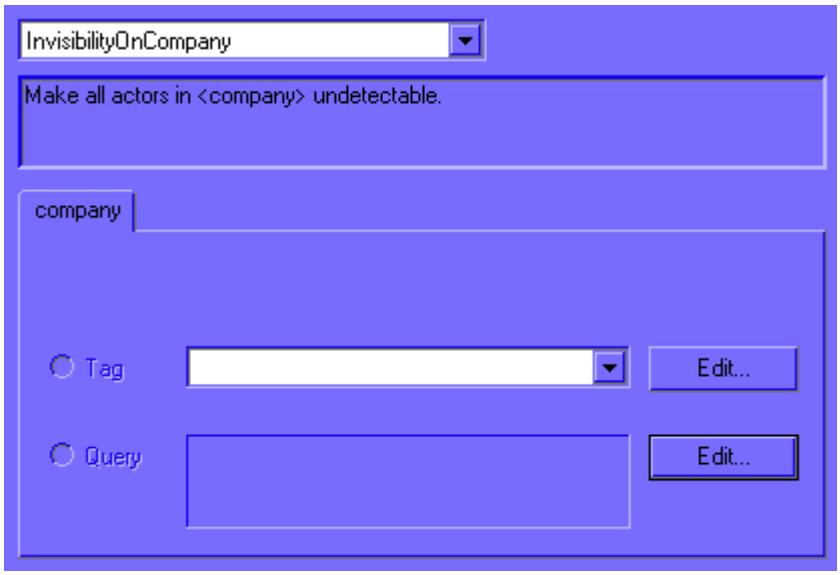
This Response turns invisibility off for Actors in the designated Team. Its only Parameter is Team. Note that invisibility doesn't make the Actors vanish from sight – it just makes other Actors unable to detect them.

InvisibiltyOnActor



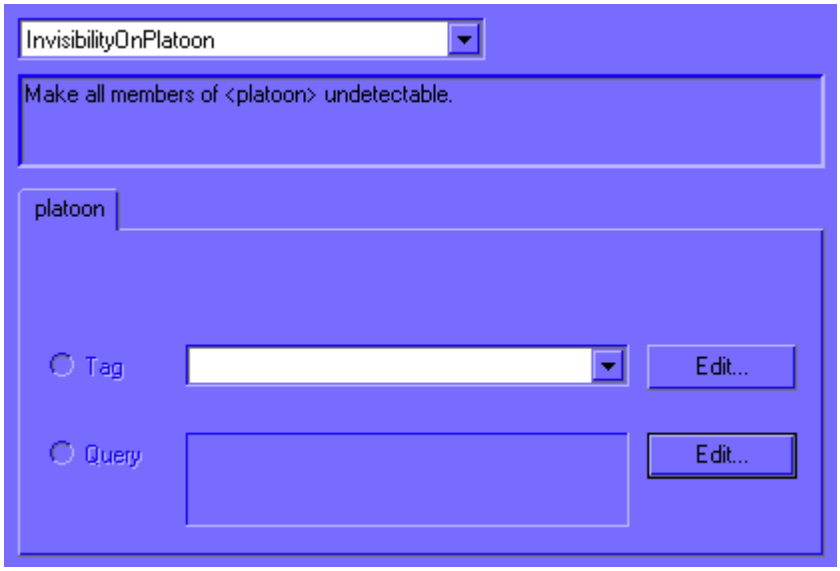
This Response turns invisibility on for the designated Actor. Its only Parameter is Actor. Note that invisibility doesn't make the Actors vanish from sight – it just makes other Actors unable to detect them.

InvisibiltyOnCompany



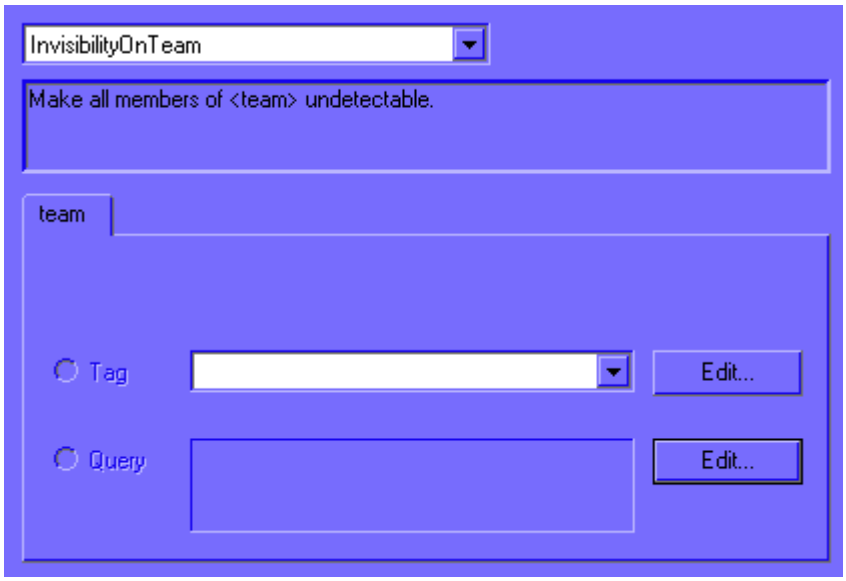
This Response turns invisibility on for Actors in the designated Company. Its only Parameter is Company. Note that invisibility doesn't make the Actors vanish from sight – it just makes other Actors unable to detect them.

InvisibiltyOnPlatoon



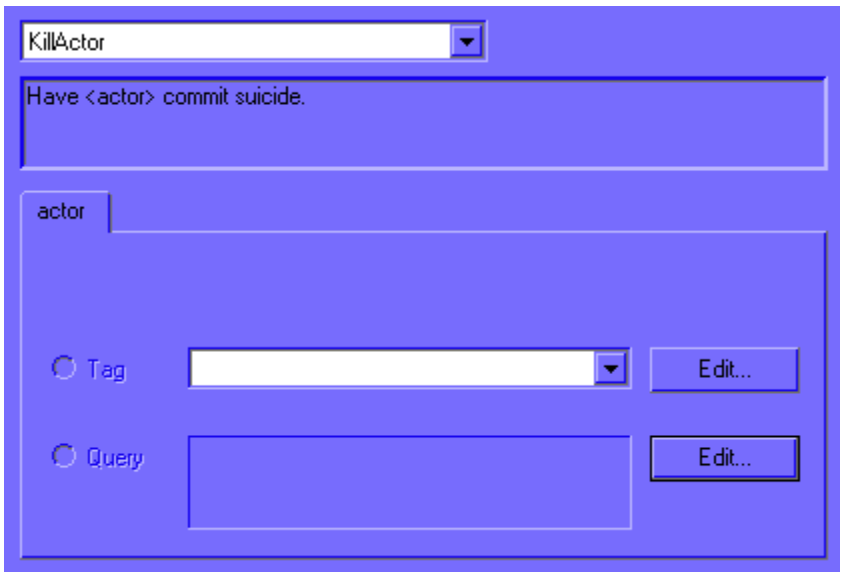
This Response turns invisibility on for Actors in the designated Platoon. Its only Parameter is Platoon. Note that invisibility doesn't make the Actors vanish from sight – it just makes other Actors unable to detect them.

InvisibiltyOnTeam



This Response turns invisibility on for Actors in the designated Team. Its only Parameter is Team. Note that invisibility doesn't make the Actors vanish from sight – it just makes other Actors unable to detect them.

KillActor

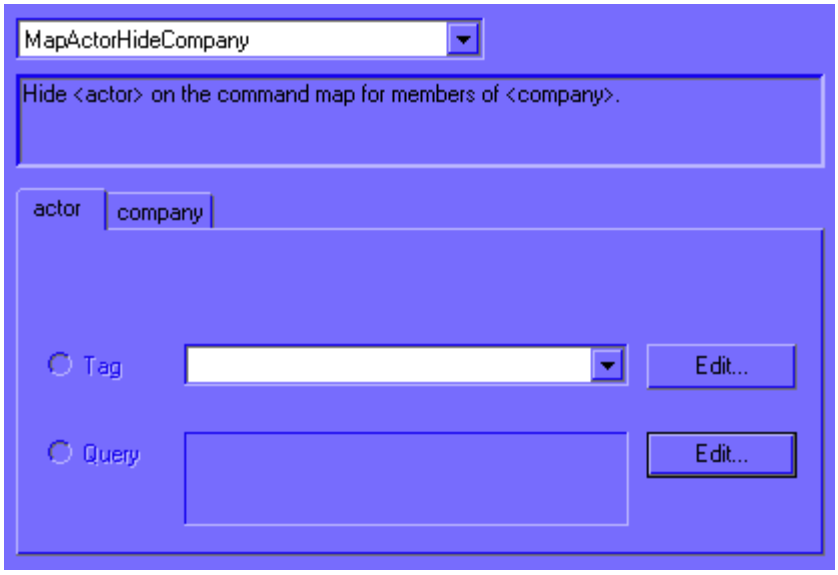


This Response causes the designated Actor to take a fatal gunshot wound to the head. Its only Parameter is Actor.

MapActorHide

This Response causes the designated Actor to vanish from the command map in the action phase of GR. Its only Parameter is Actor. This Response is usually used in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

MapActorHideCompany

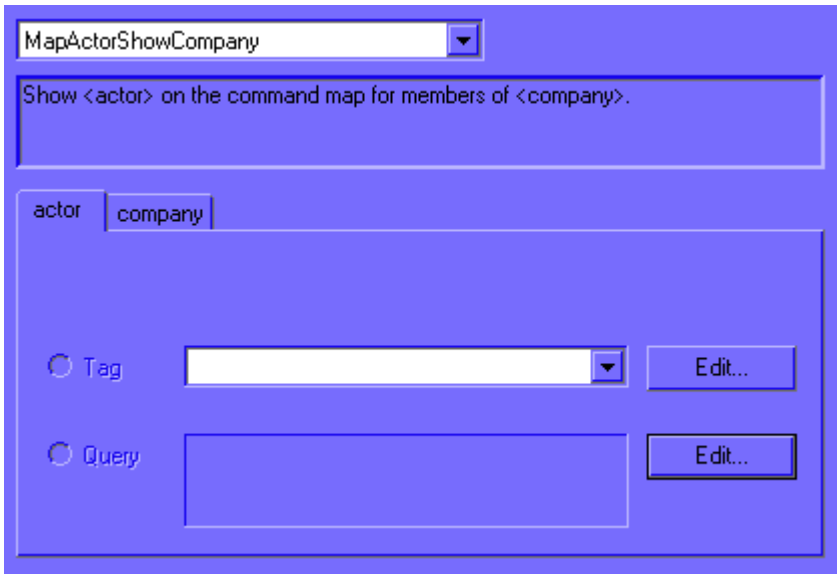


This Response causes the designated Actor to vanish from the command map in the action phase of GR for to the specified Company. Its Parameters are Actor and Company. This Response is usually used in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

MapActorShow

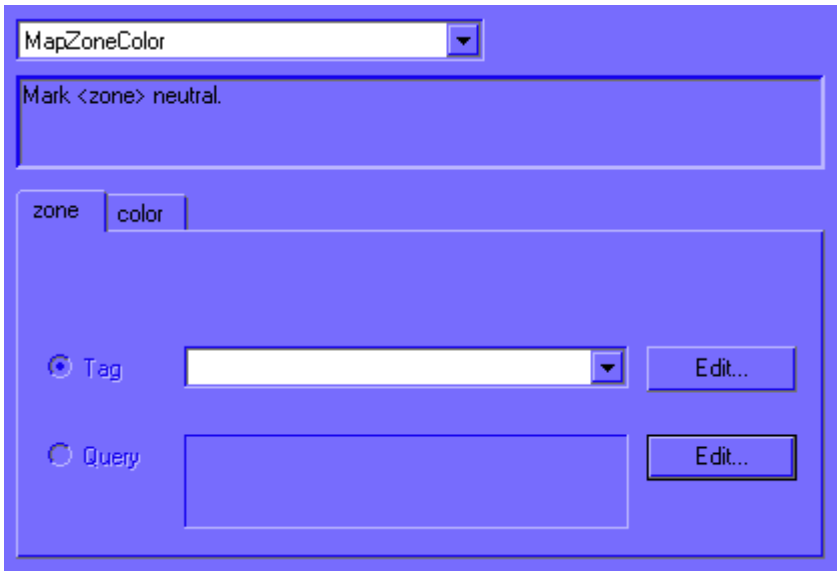
This Response forces the designated Actor to appear on the command map for all players in the action phase of GR, even if that Actor would not normally show up on a player's command map. Its only Parameter is Actor. This Response is usually used in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

MapActorShowCompany



This Response forces the designated Actor to appear on the command map for all players in the specified Company in the action phase of GR, even if that Actor would not normally show up on a player's command map. Its Parameters are Actor and Company. This Response is usually used in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

MapZoneColor

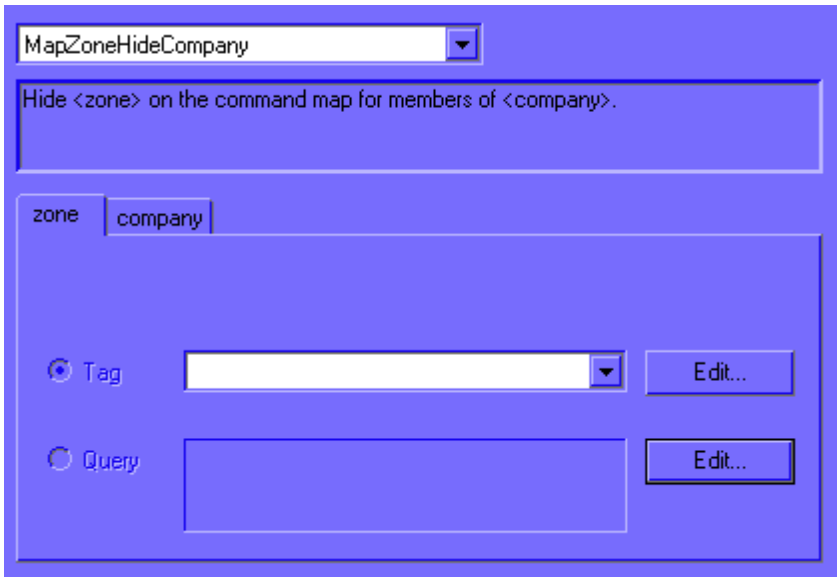


This Response sets the specified Zone to emit smoke of the designated color. The Parameters are Zone and Color. This Response is used mainly in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

MapZoneHide

This Response hides the specified Zone from the command map for all players. The only Parameter is Zone. This Response is used mainly in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

MapZoneHideCompany

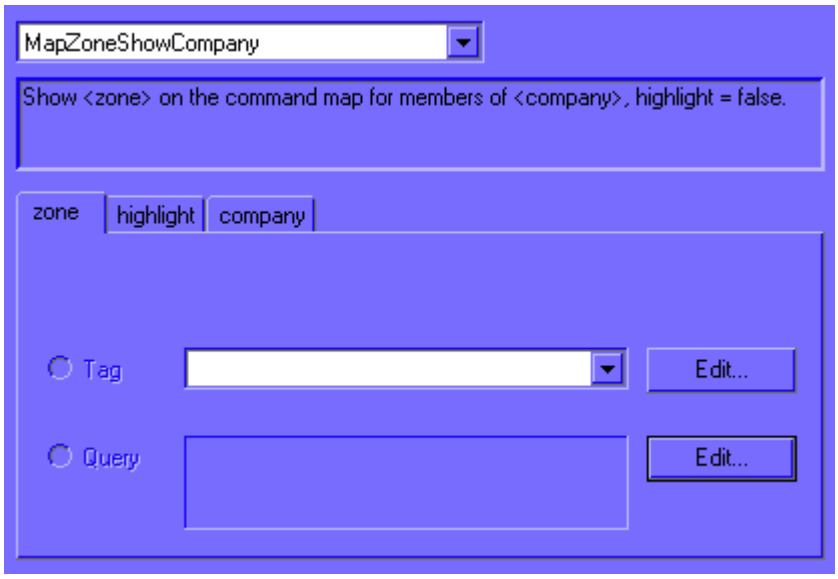


This Response hides the specified Zone from the command map for all players in the designated Company. Its Parameters are Zone and Company. This Response is used mainly in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

MapZoneShow

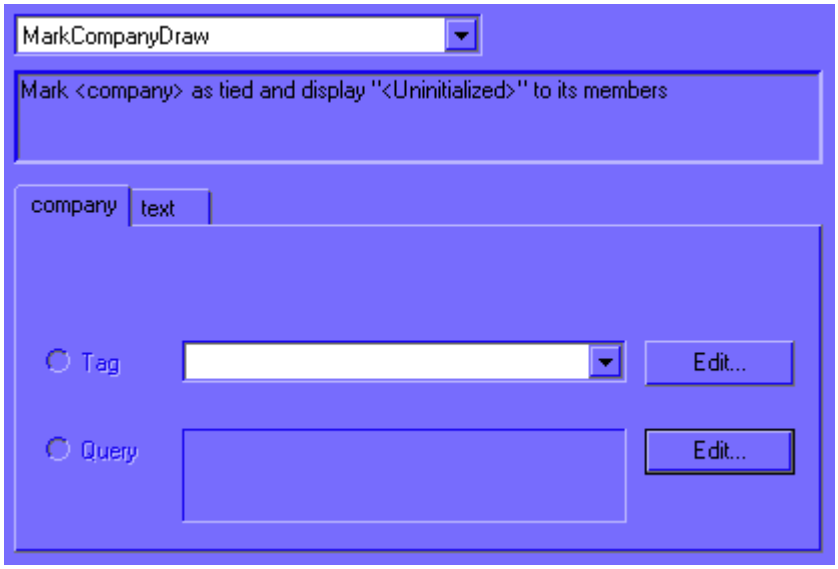
This Response makes the specified Zone appear on the command map for all players. Its Parameters are Zone and Highlight, and if Highlight is true the Zone will emit smoke. This Response is used mainly in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

MapZoneShowCompany



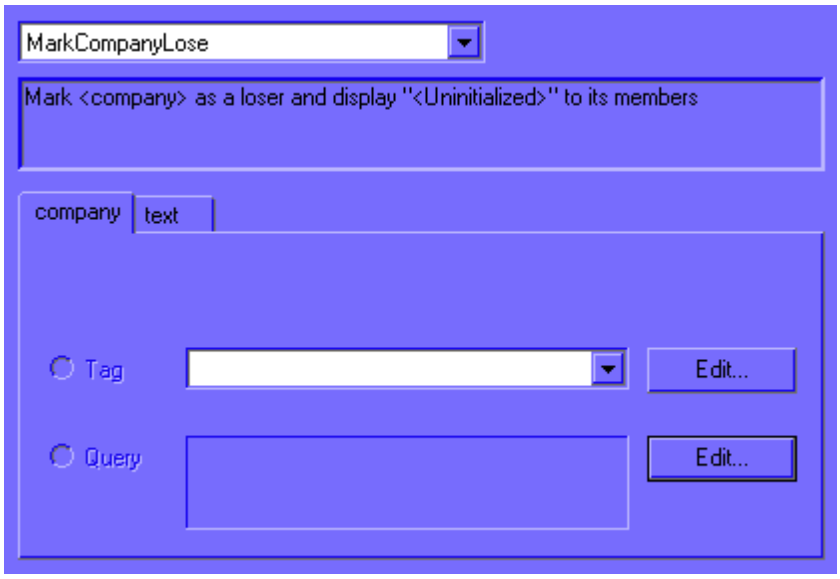
This Response makes the specified Zone appear on the command map for all players in the designated Company. Its Parameters are Zone, Highlight, and Company. If Highlight is true the Zone will emit smoke. This Response is used mainly in multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information.

MarkCompanyDraw



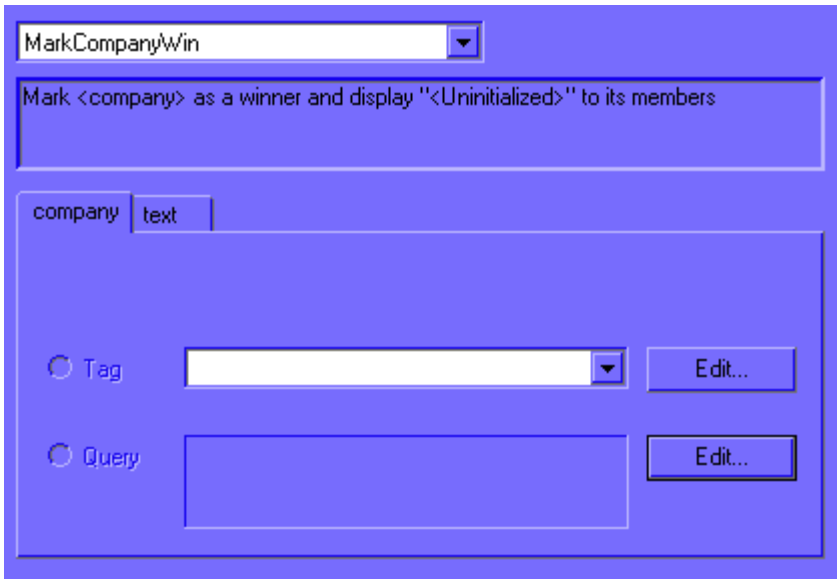
This Response marks the designated Company as 'tied' and shows all players in that Company a user-defined message. The Parameters are Company and Text. This Response is used only in multiplayer game types. For more information refer to the In-Depth Multiplayer Scripting section.

MarkCompanyLose



This Response marks the designated Company as the loser and displays a user-defined message to the players on that Company. Its Parameters are Company and Text. This Response is used only in multiplayer game types. For more information refer to the In-Depth Multiplayer Scripting section.

MarkCompanyWin

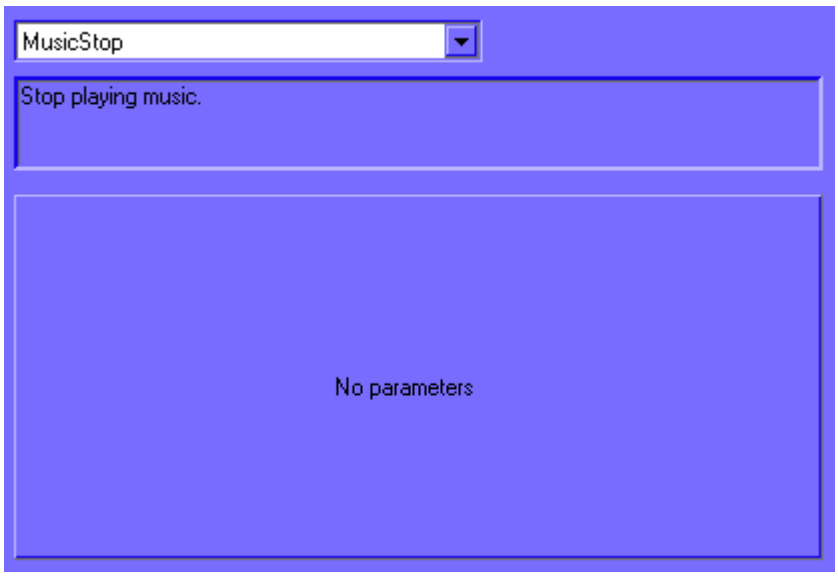


This Response marks the designated Company as the winner and displays a user-defined message to the players on that Company. Its Parameters are Company and Text. This Response is used only in multiplayer game types. For more information refer to the In-Depth Multiplayer Scripting section.

MusicStart

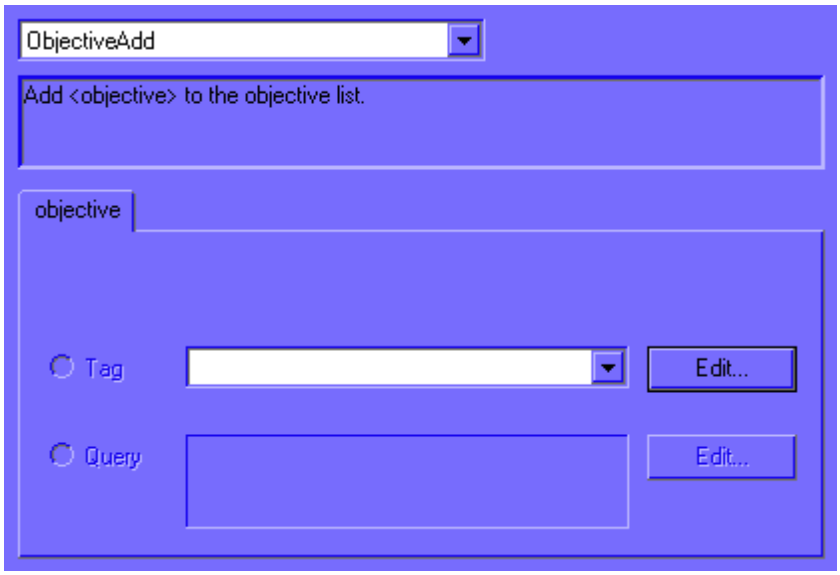
This Response causes the designated wav file to play as music, i.e. the music options in GR will affect how this music is played. Its only Parameter is Track.

MusicStop



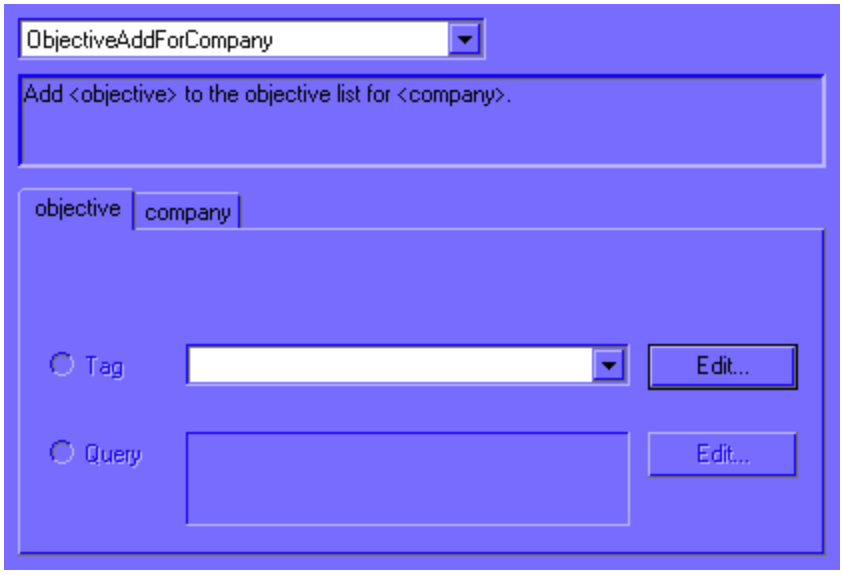
This Response causes all music to fade out and stop playing. It has no Parameters.

ObjectiveAdd



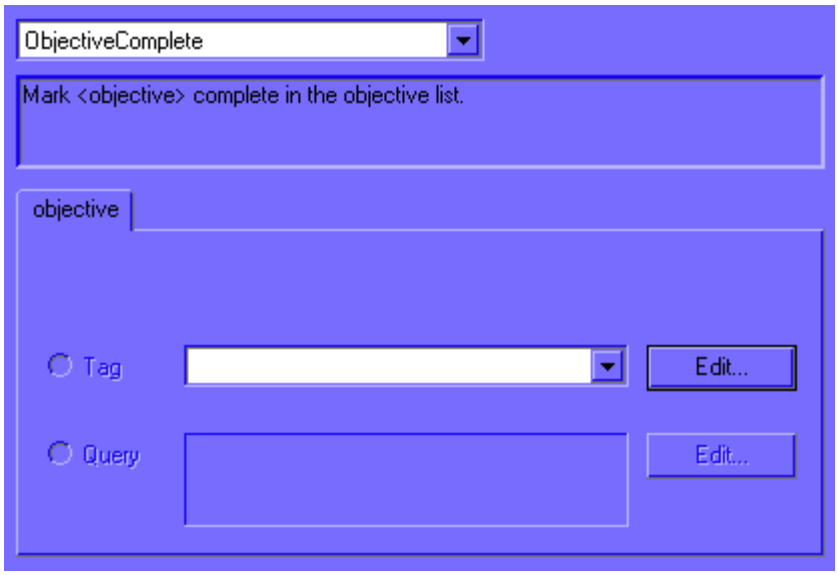
This Response adds the designated Objective to the Objective List. If the Objective is already in the Objective List, nothing happens. Its only Parameter is Objective.

ObjectiveAddforCompany



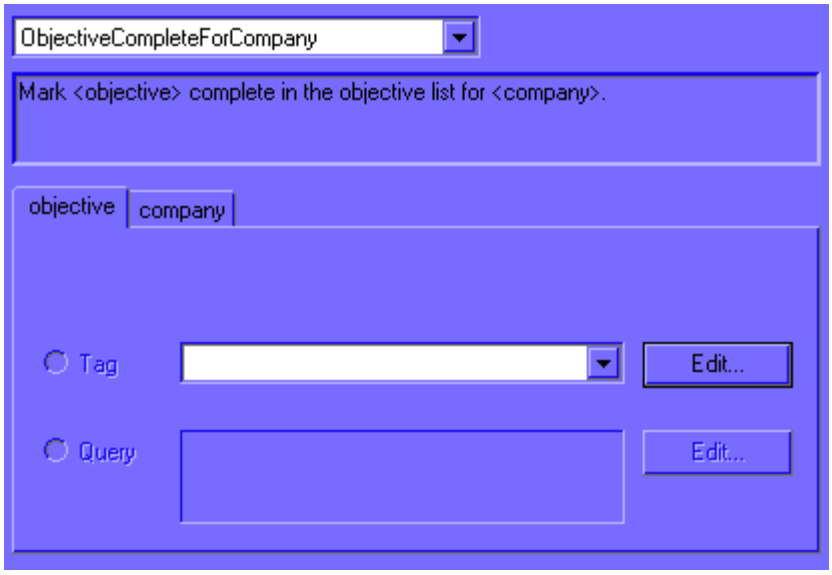
This Response adds the designated Objective to the Objective List for the specified Company. If that Company already has that Objective in the Objective List, nothing happens. Its Parameters are Objective and Company.

ObjectiveCompl;ete



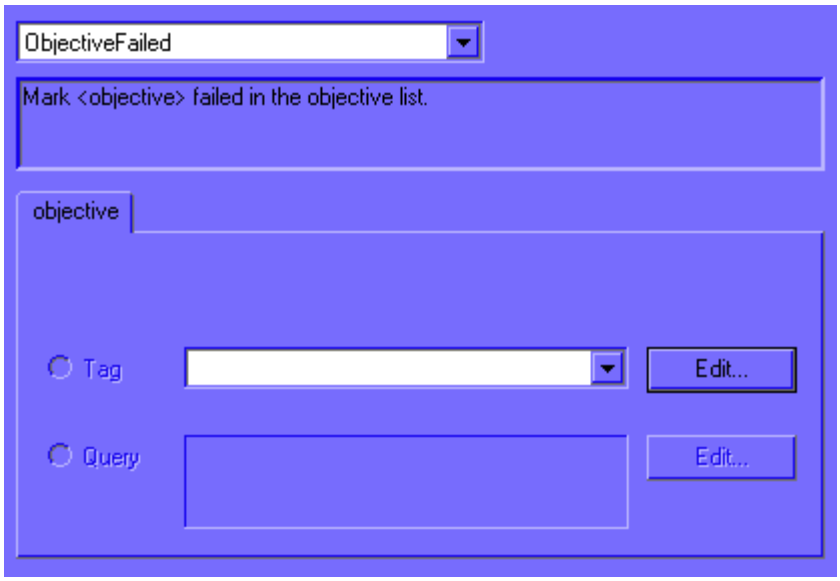
This Response marks the designated Objective as complete. Its Parameter is Objective.

ObjectiveCompleteforCompany



This Response marks the designated Objective as complete for the specified Company. Its Parameters are Objective and Company.

ObjectiveFailed

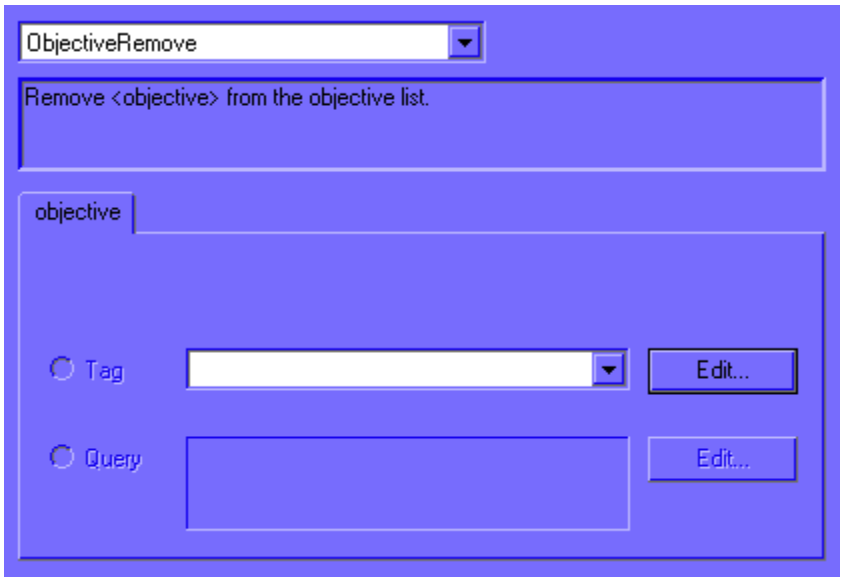


This Response marks the designated Objective as failed. Its only Parameter is Objective.

ObjectiveFailedforCompany

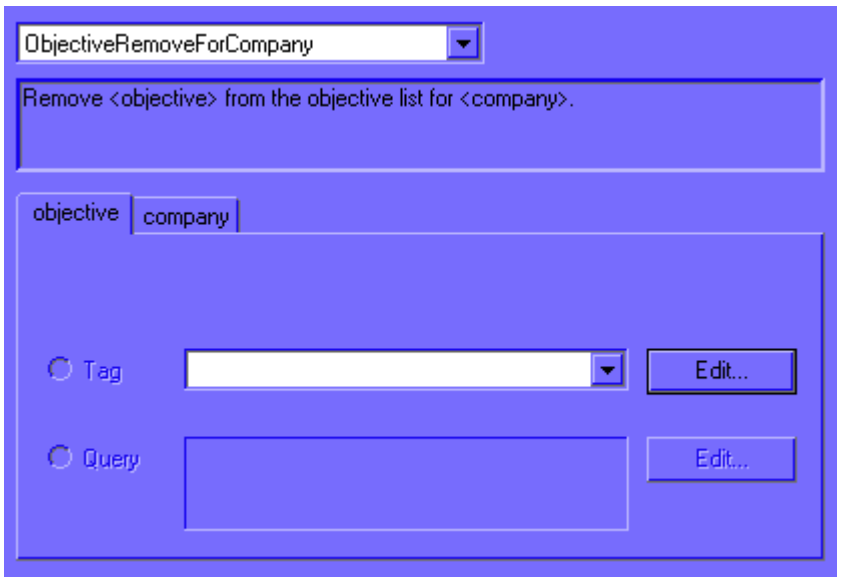
This Response marks the designated Objective as failed for the specified Company. Its Parameters are Objective and Company.

ObjectiveRemove



This Response removes the designated Objective from the Objective List. Its only Parameter is Objective.

ObjectiveRemoveforCompany

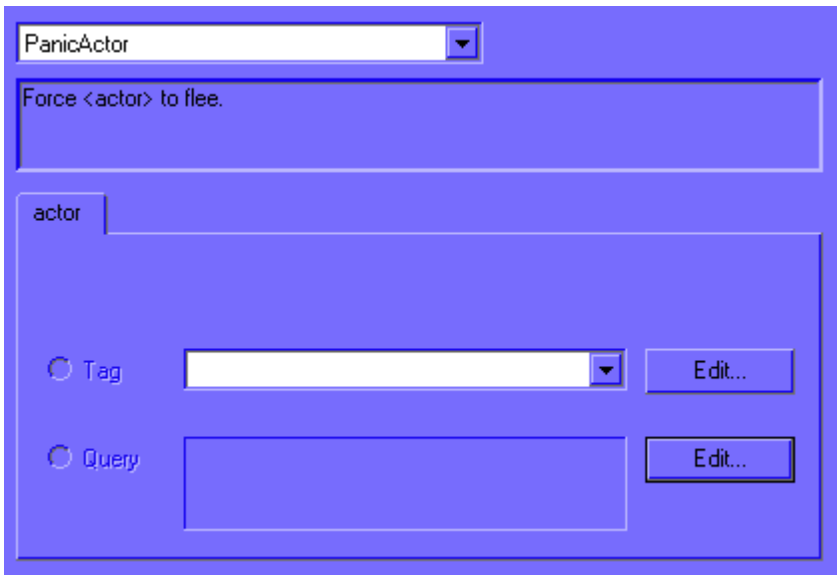


This Response removes the designated Objective from the Objective List for the specified Company. Its Parameters are Objective and Company.

OpenDoor

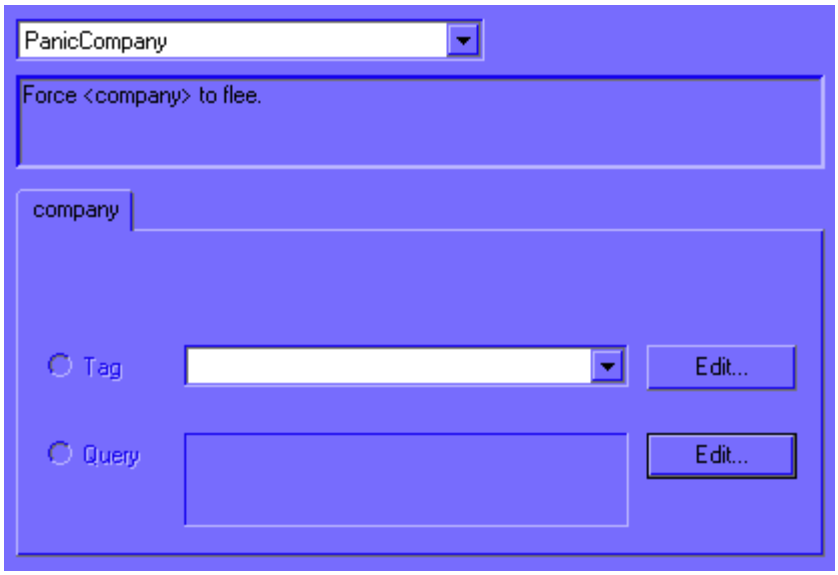
This Response opens the designated Door. If the Door is already open, nothing will happen. Its only Parameter is Door.

PanicActor



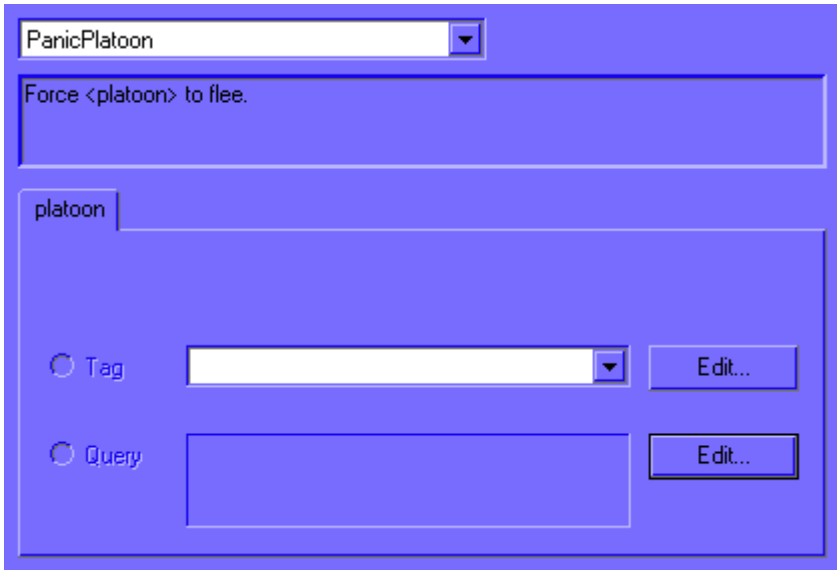
This Response forces an Actor into the panic/surrender behavior. Its only Parameter is Actor.

PanicCompany



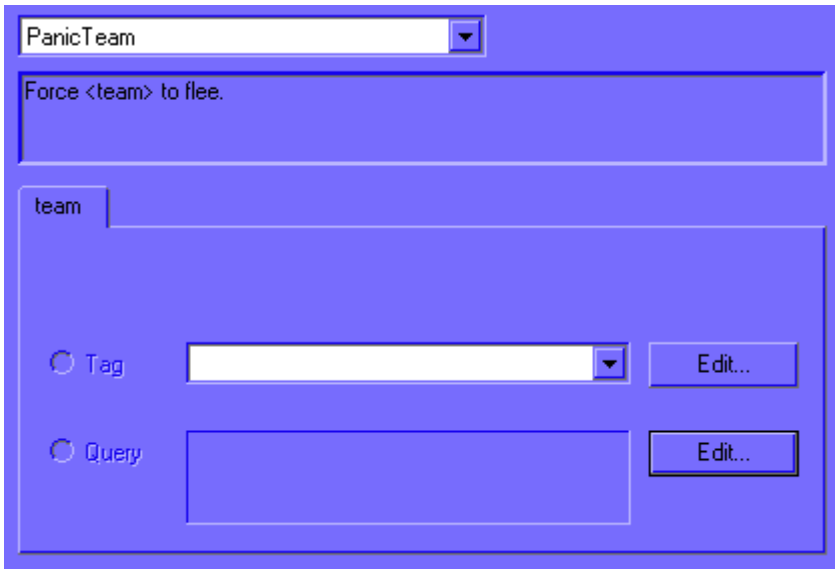
This Response forces all Actors in the designated Company into the panic/surrender behavior. Its only Parameter is Company.

PanicPlatoon



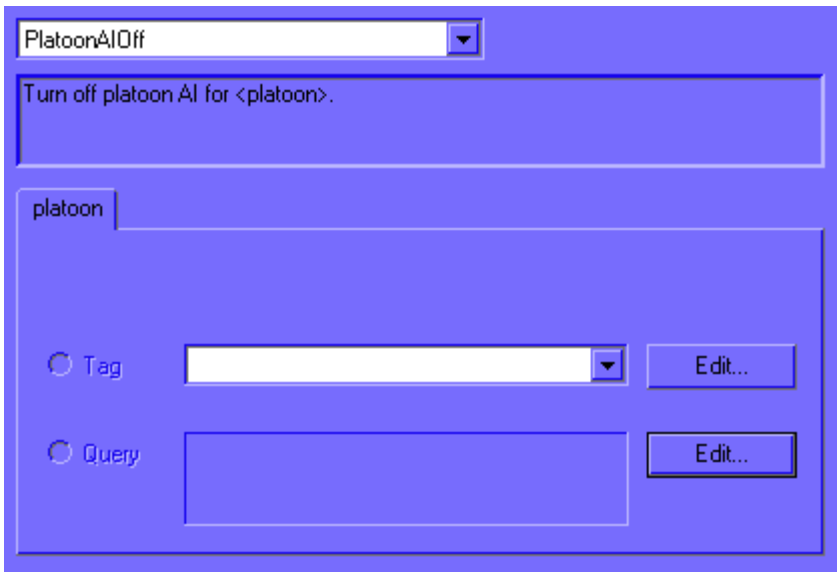
This Response forces all Actors in the designated Platoon into the panic/surrender behavior. Its only Parameter is Platoon.

PanicTeam



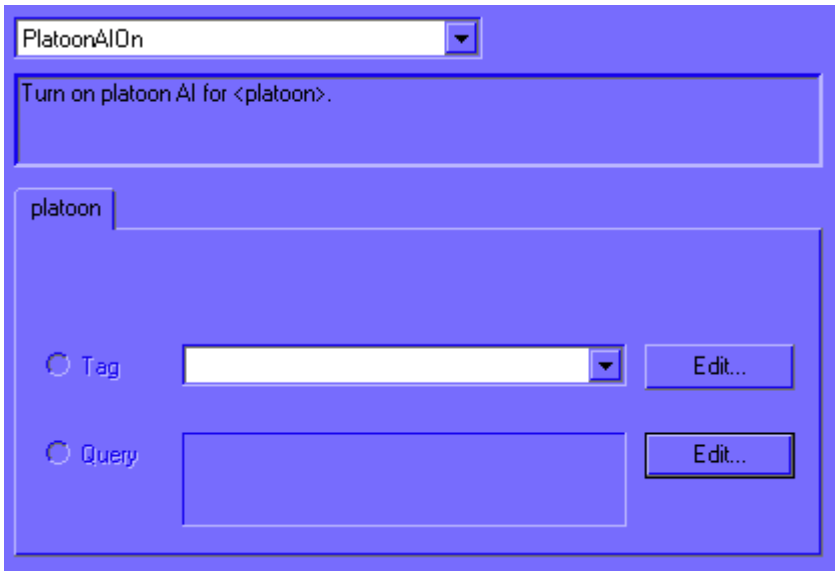
This Response forces all Actors in the designated Team into the panic/surrender behavior. Its only Parameter is Team.

PlatoonAIOff



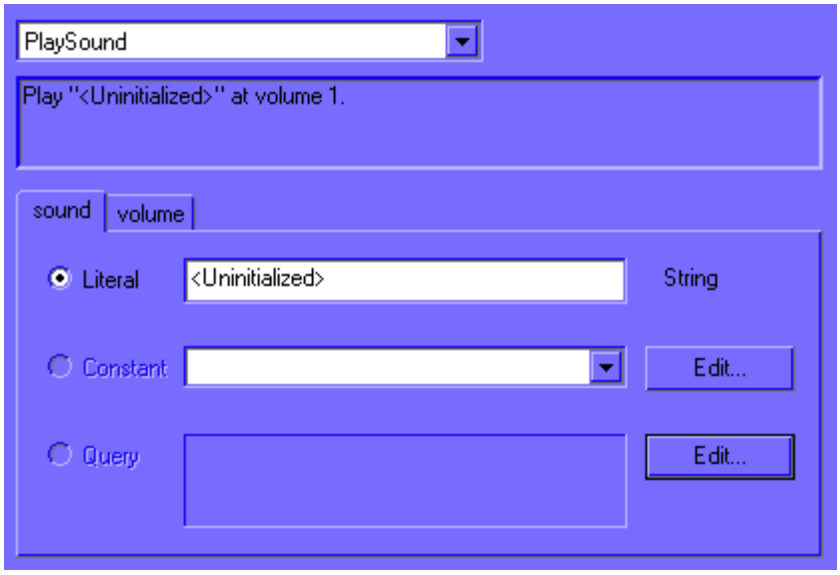
This Response turns off Platoon-level AI for the designated Platoon. Its only Parameter is Platoon.

PlatoonAIOn



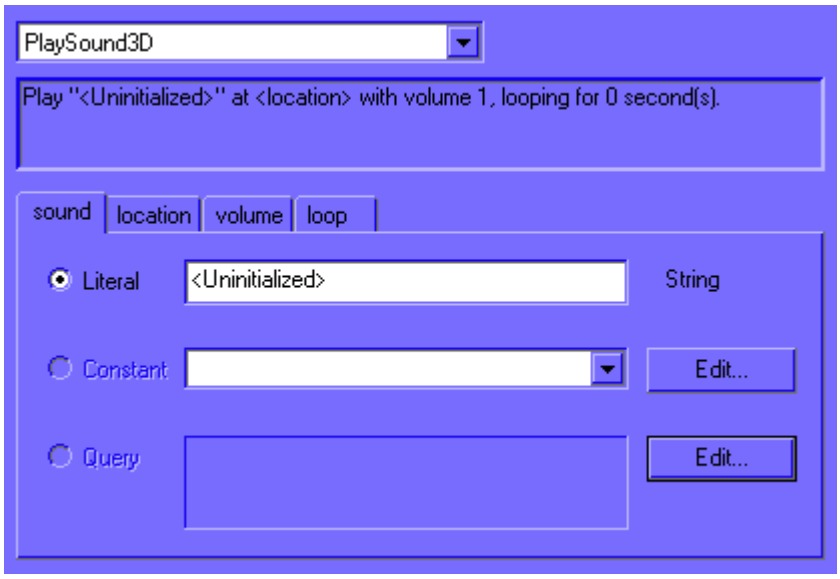
This Response turns on Platoon-level AI for the designated Platoon. Platoon-level AI is on by default. Its only Parameter is Platoon.

PlaySound



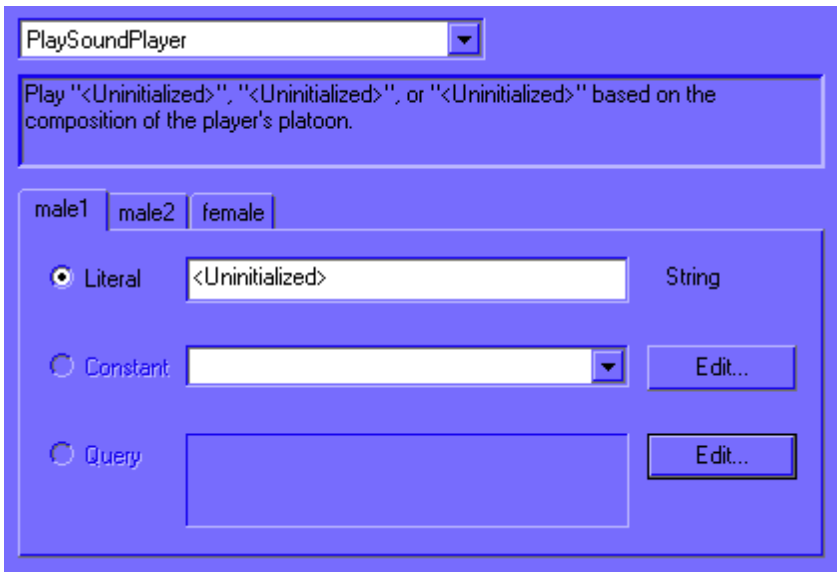
This Response will play the designated wav file as a 2d sound to all players. Its Parameters are Sound and Volumen.

PlaySound3d



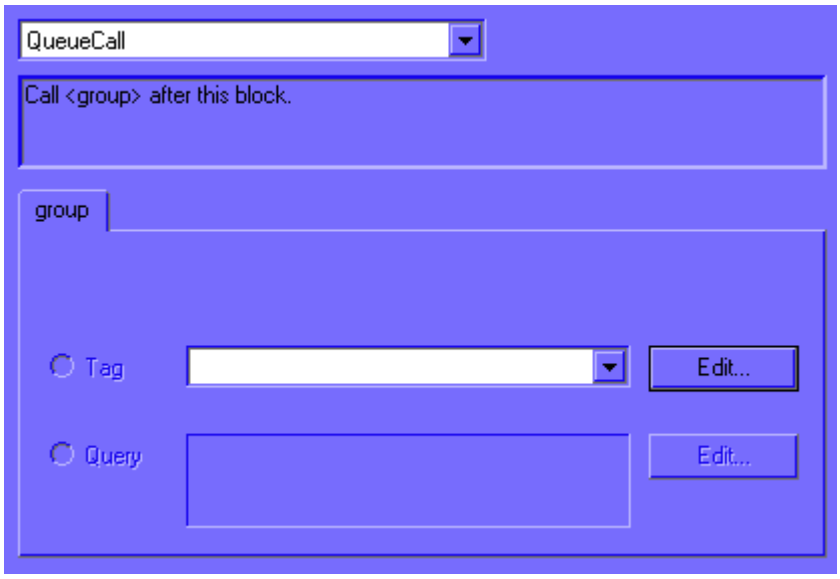
This Response plays the designated wav file at the specified location and the user-defined volume, and it can be made to loop for a set number of seconds. Its Parameters are Sound, Location, Volume, and Loop.

PlaySoundPlayer



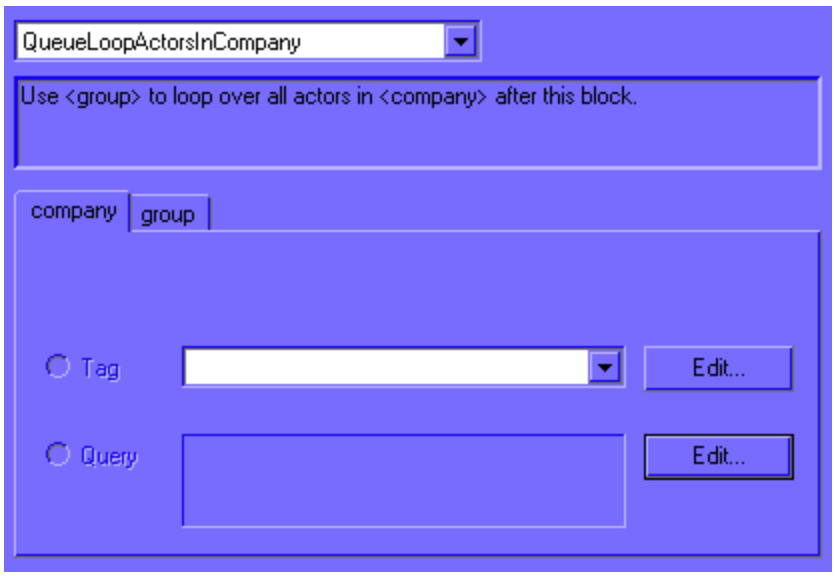
This Response plays the designated wav files as 2d sounds depending on the composition of the player's Platoon. The Parameters are Male1, Male2, and Female.

QueueCall



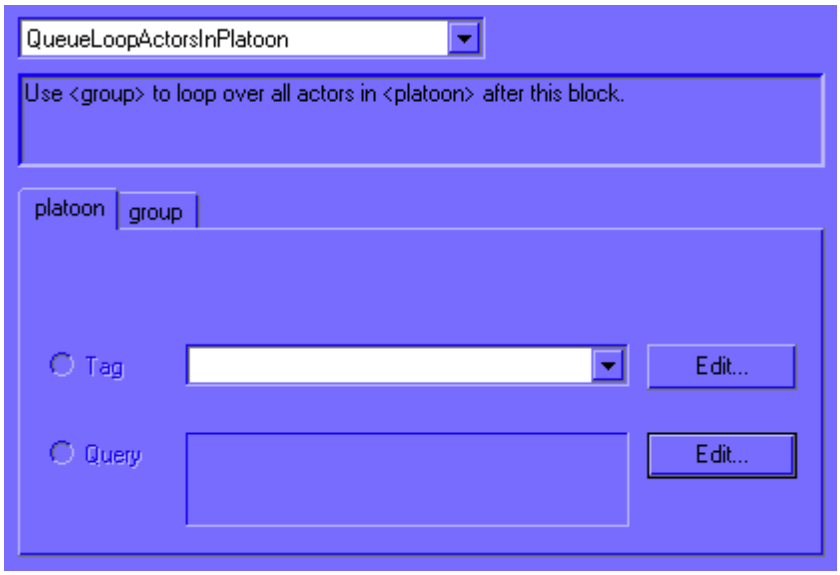
This Response forces the blocks in the designated Group to execute if those blocks use the "Call" Trigger Event. Its only Parameter is Group.

QueueLoopActorsinCompany



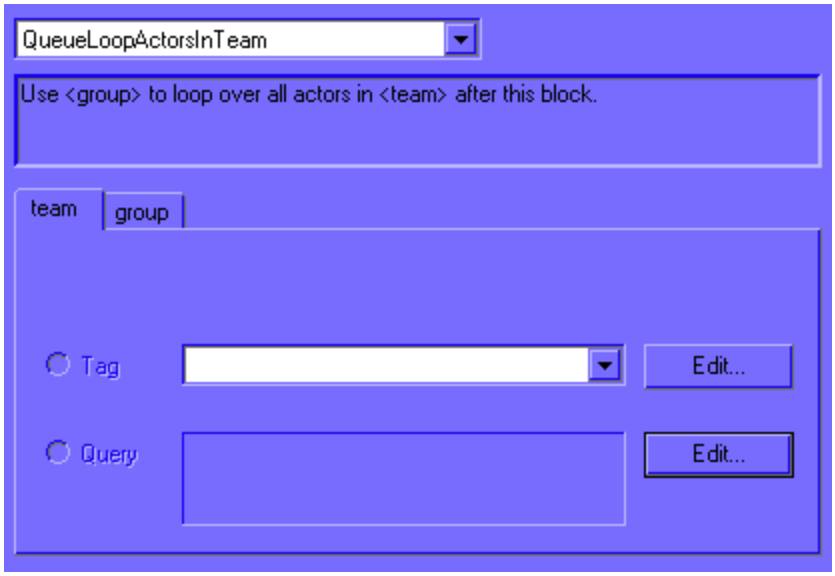
This Response uses the block in the designated Group to initiate a loop through all of the Actors in the specified Company after this block finishes executing. Its Parameters are Group and Company.

QueueLoopActorsinPlatoon



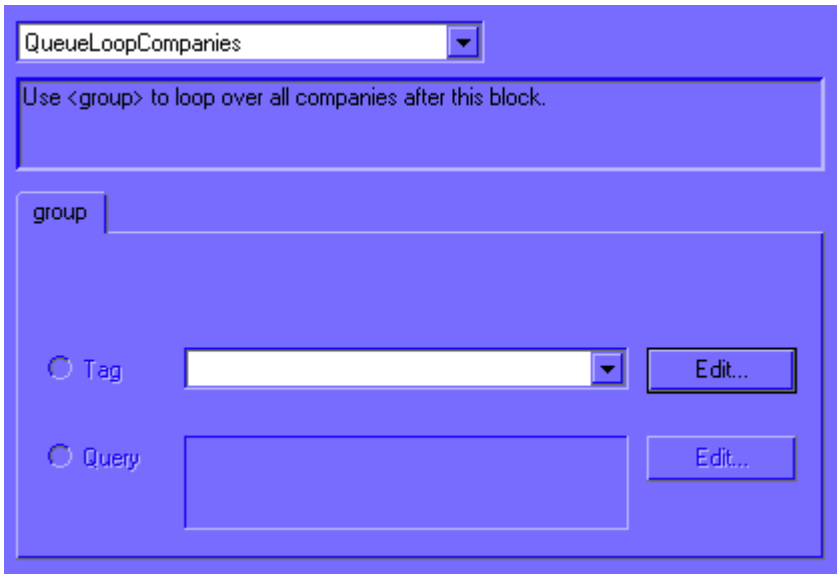
This Response uses the block in the designated Group to initiate a loop through all of the Actors in the specified Platoon after this block finishes executing. Its Parameters are Group and Platoon.

QueueLoopActorsinPlatoon



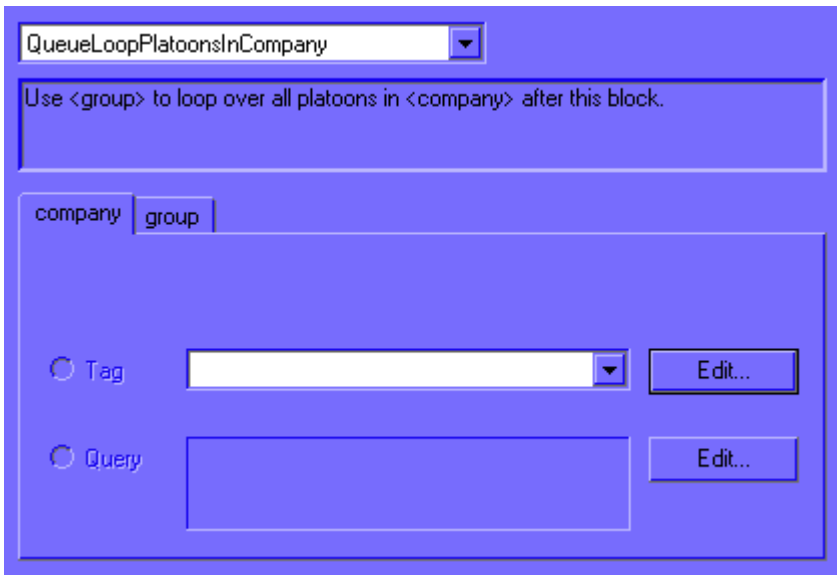
This Response uses the block in the designated Group to initiate a loop through all of the Actors in the specified Team after this block finishes executing. Its Parameters are Group and Team.

QueueLoopCcompanies



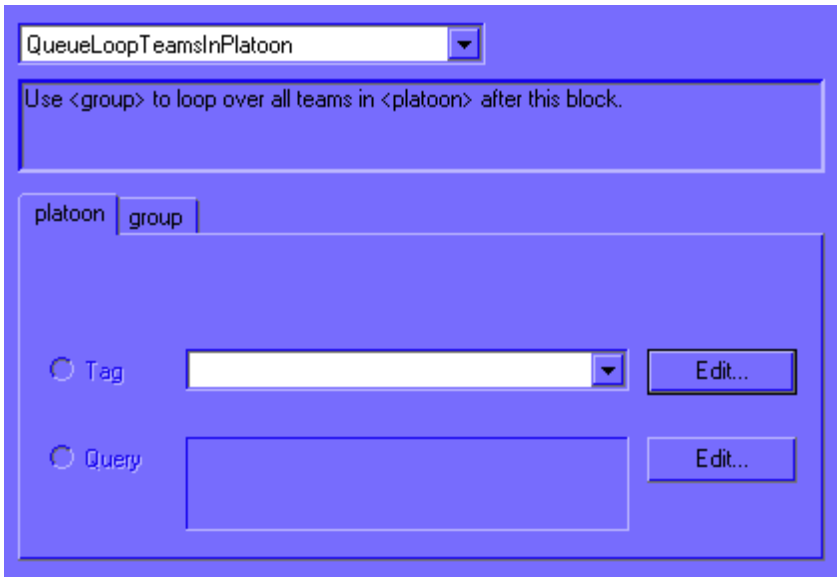
This Response uses the block in the designated Group to initiate a loop through all of the Companies currently in the mission after this block finishes executing. Its Parameters are Group and Company.

QueueLoopPlatoonsinCompany



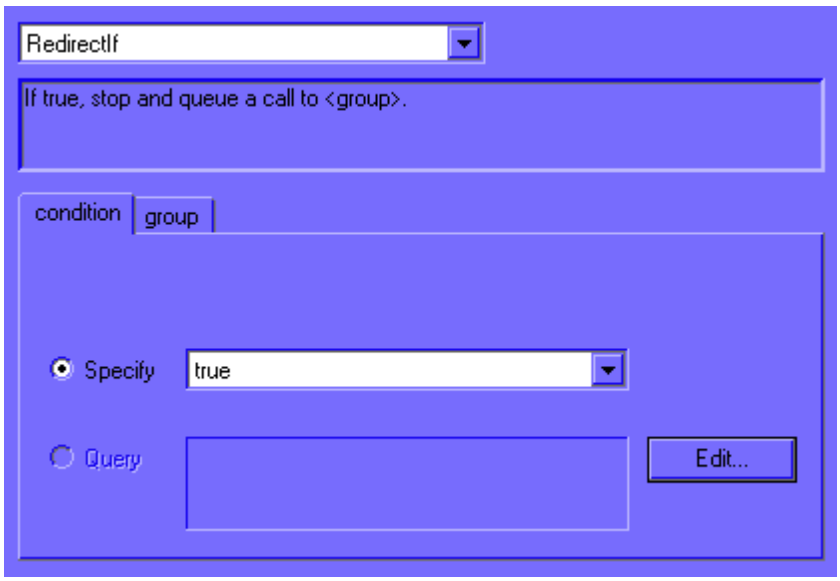
This Response uses the block in the designated Group to initiate a loop through all Platoons in the specified Company after this block finishes executing. Its Parameters are Group and Company.

QueueLoopTeamsinPlatoon



This Response uses the block in the designated Group to initiate a loop through all Teams in the specified Platoon after this block finishes executing. Its Parameters are Group and Company.

RedirectIf

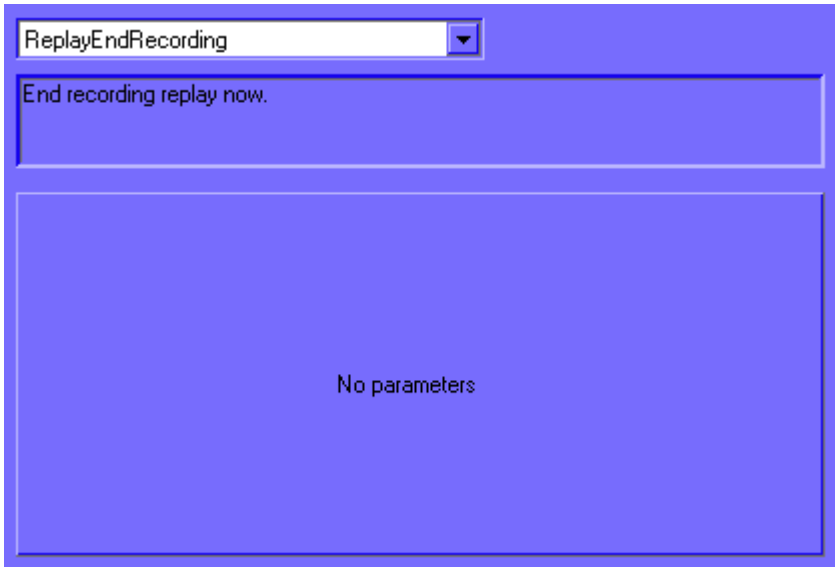


This Response will call the blocks in a Group that are Triggered with Call if the designated conditions are true. Its Parameters are Condition and Group.

ReplayBeginRecording

This Response forces the replay to begin recording if the player has replays activated in his GR options. It has no Parameters. Note that this Response also activates the mission replay save, so if the mission is replayed by choosing "Replay" from a menu, it will begin from the point the replay recording started. Also note that the replay should be overridden with the ReplayOverride Response (see below) if the user doesn't want the replay to start from the mission start before using this Response.

ReplayEndRecording

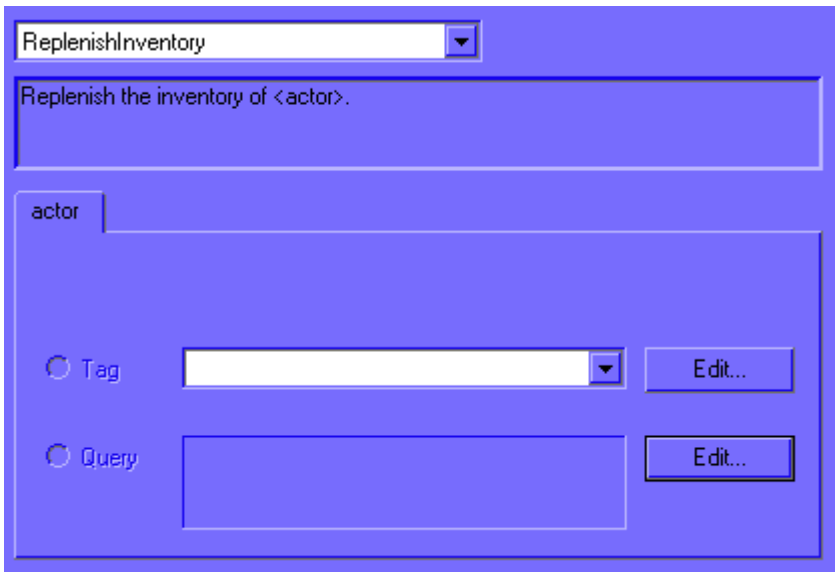


This Response forces the replay recording to stop. It has no Parameters. It should only be used at the end of replay. It has no Parameters. To make the replay start later, see "ReplayOverride", below, and "ReplayBeginRecordi ng" above.

ReplayOverride

This Response stops the replay recording until the ReplayBeginRecording Response (see above) is used. It has no Parameters.

ReplenishInventory

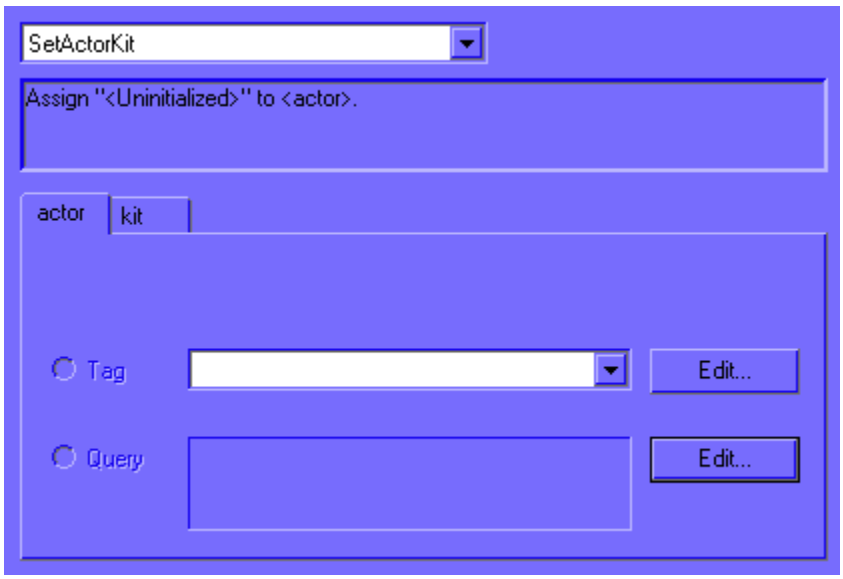


This Response completely refills the inventory of the designated Actor. Its only Parameter is Actor.

ResetMapObject

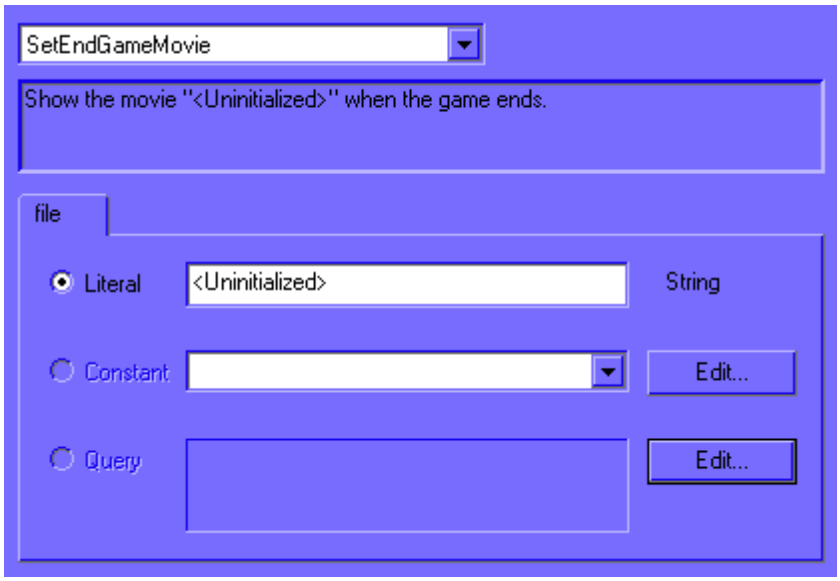
This Response forces a map object back into its original state. Its only Parameter is Object. Note that it was intended only for the targets in the Training section of GR, and has not been tested with any other map object.

SetActorKit



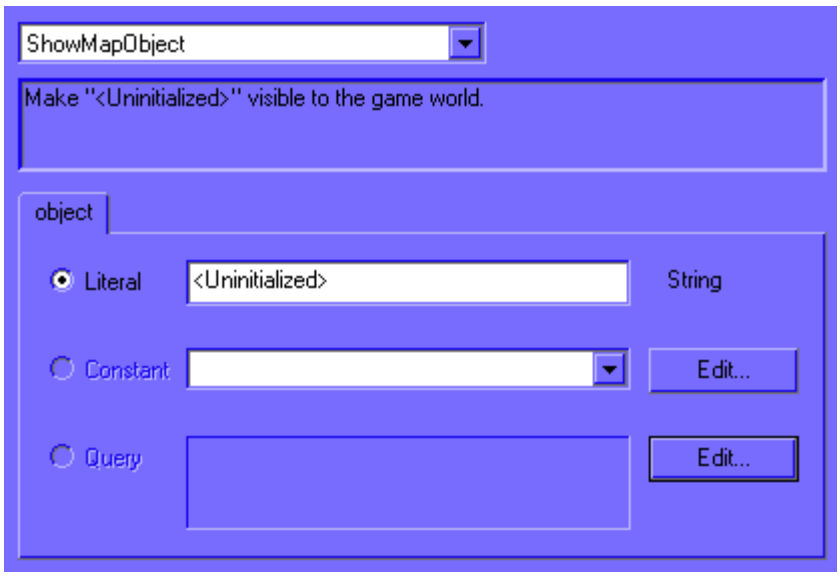
This Response gives the designated kit to the specified Actor. Its Parameters are Actor and Kit.

SetEndGameMovie



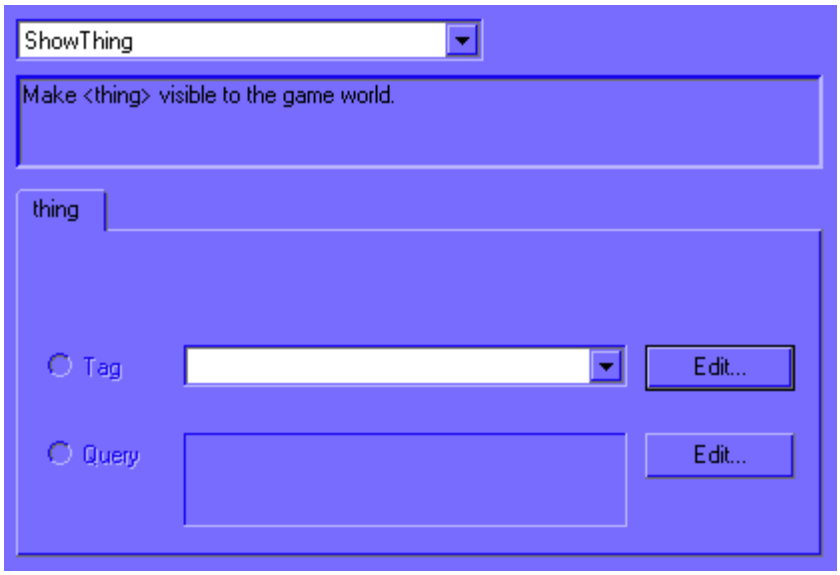
This Response will make the game play the designated avi file after the current mission successfully ends. Its only Parameter is File.

ShowMapObject



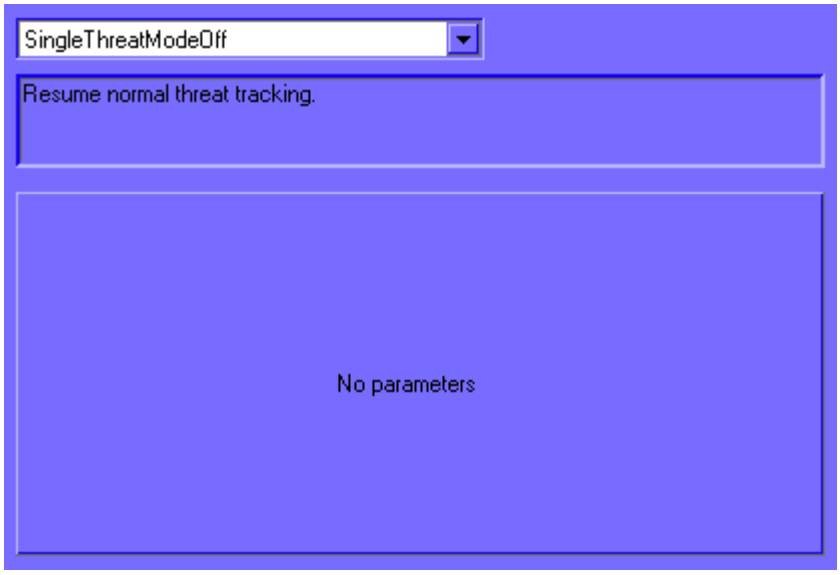
This Response makes a map object that has been hidden with HideMapObject visible. The only Parameter is Object.

ShowThing



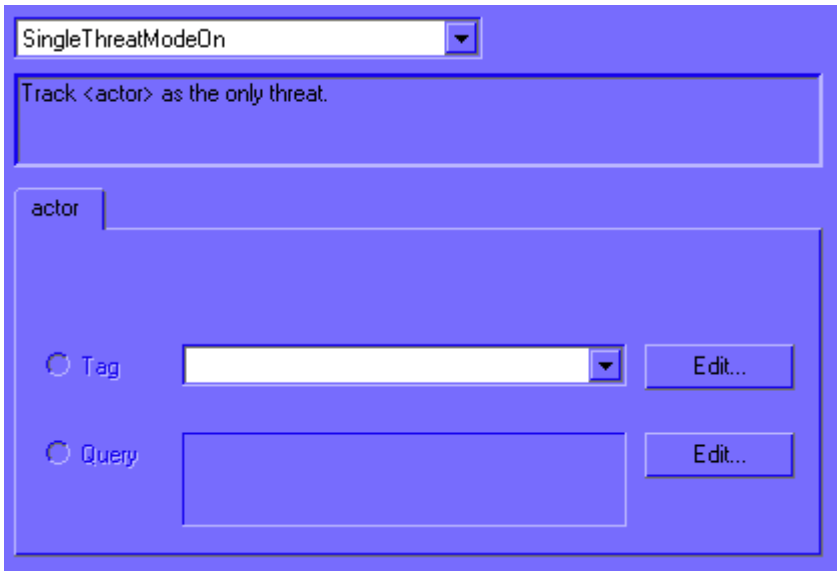
This Response will make an Element hidden by the HideThing Response visible. The only Parameter is Thing.

SingleThreatModeOff



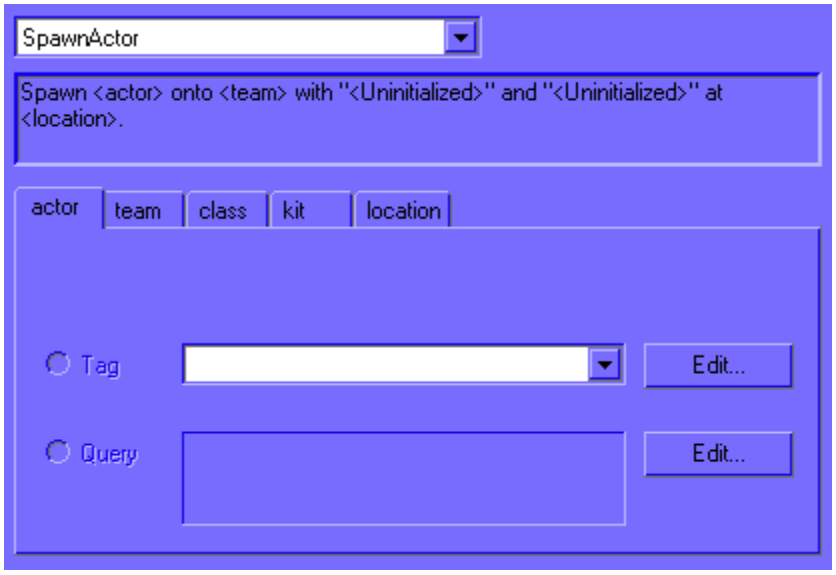
This Response returns the threat indicator to normal after it has been changed by the SingleThreatModeOn Response (see below). It has no Parameters.

SingleThreatModeOn



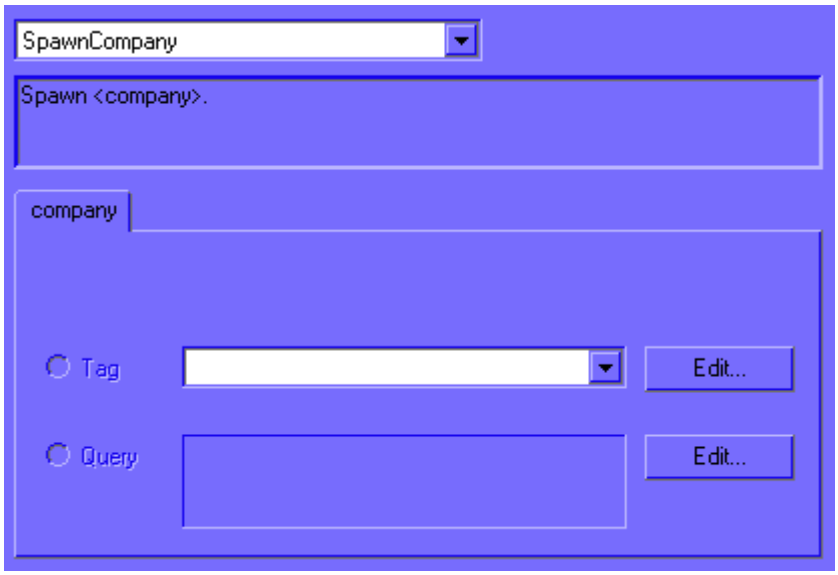
This Response forces the threat indicator to only track the designated Actor. Its only Parameter is Actor.

SpawnActor



This Response will spawn an Actor onto a Team with the designated class and kit at the specified location. It has the Actor, Team, Class, Kit, and Location Parameters. Note that this is mainly used for multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information. Also note that this Response is only meant to be used in a block triggered by the PreAction Trigger Event.

SpawnCompany

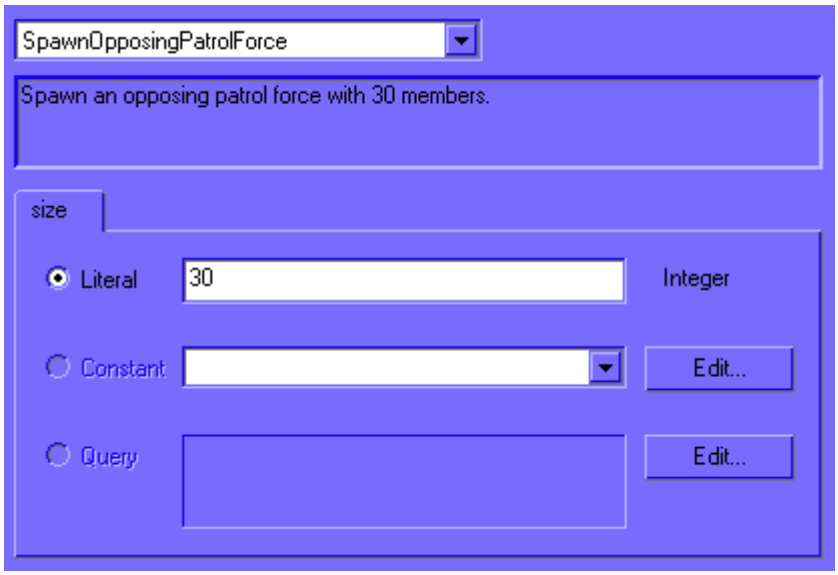


This Response spawns the designated Company. This Company will be empty unless the other spawn Responses are used. Its only Parameter is Company. Note that this is mainly used for multiplayer game types. Refer to the In-Depth Multiplayer Scripting section for more information. Also note that this Response is only meant to be used in a block triggered by the PreAction Trigger Event.

SpawnOpposingAssaultForce

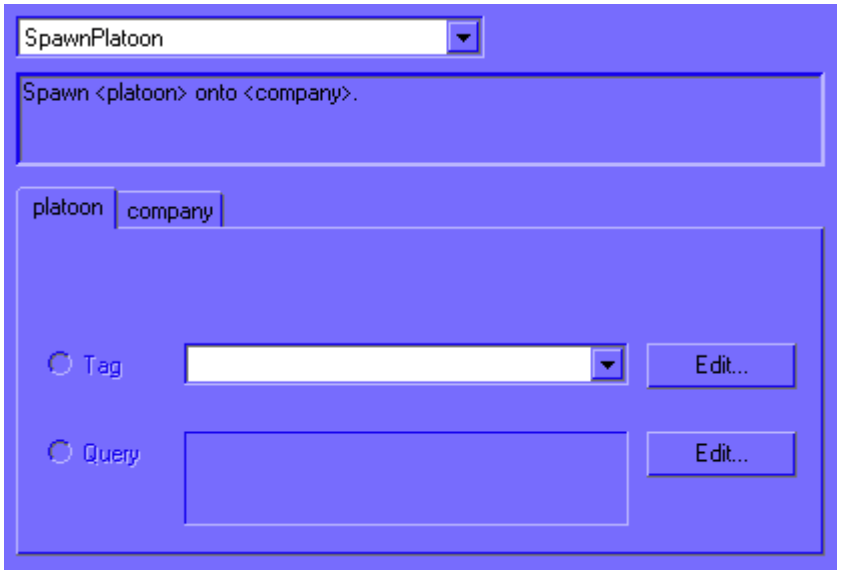
This Response will spawn a new company of enemies consisting of the designated number of Actors, all of whom will use the Opposing Force kit and Actor files. These enemies will spawn at the Zones with "Assault Platoon" checked in the Zone Properties window. They will be given an automatic time-delay plan to assault the Zone with "SP/Coop Base" checked in the Zone Properties window, if a zone like that exists. Its only Parameter is Size. Note that this Response is only meant to be used in a block triggered by the PreAction Trigger Event.

SpawnOpposingPatrolForce



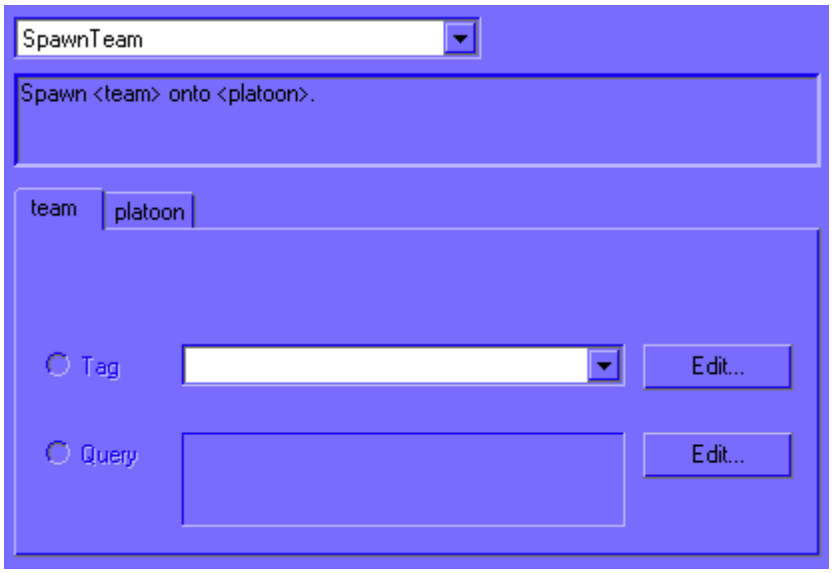
This Response will spawn a new company of enemies consisting of the designated number of Actors, all of whom will use the Opposing Force kit and Actor files. These enemies will spawn at Zones with "Point" checked in the Zone Properties Window if any are available, and they will automatically patrol their immediate area. Its only Parameter is Size. Note that this Response is only meant to be used in a block triggered by the PreAction Trigger Event.

SpawnPlatoon



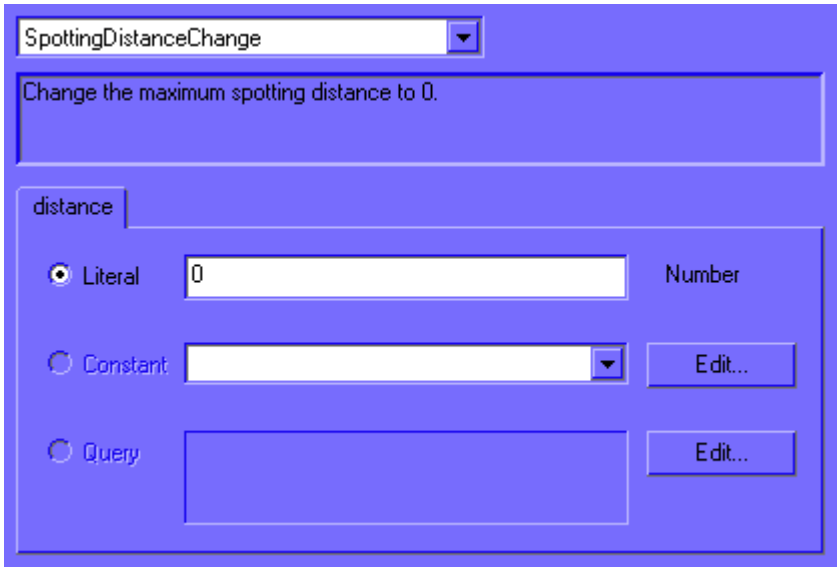
This Response will spawn the designated Platoon into the designated Company. This Platoon will be empty unless other spawn Responses are used. Its Parameters are Platoon and Company. Note that this Response is only meant to be used in a block triggered by the PreAction Trigger Event.

SpawnTeam



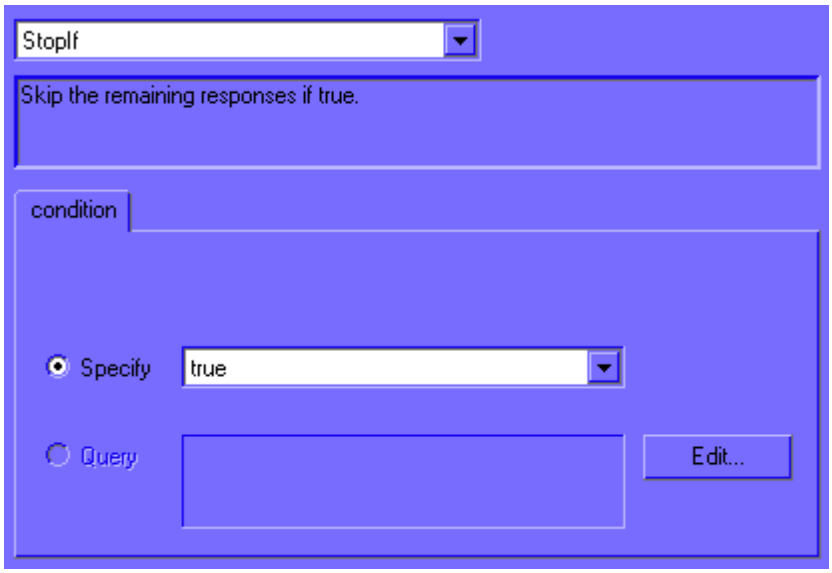
This Response will spawn the designated Team into the designated Platoon. This Team will be empty unless other spawn Responses are used. Its Parameters are Team and Platoon. Note that this Response is only meant to be used in a block triggered by the PreAction Trigger Event.

SpottingDistanceChange



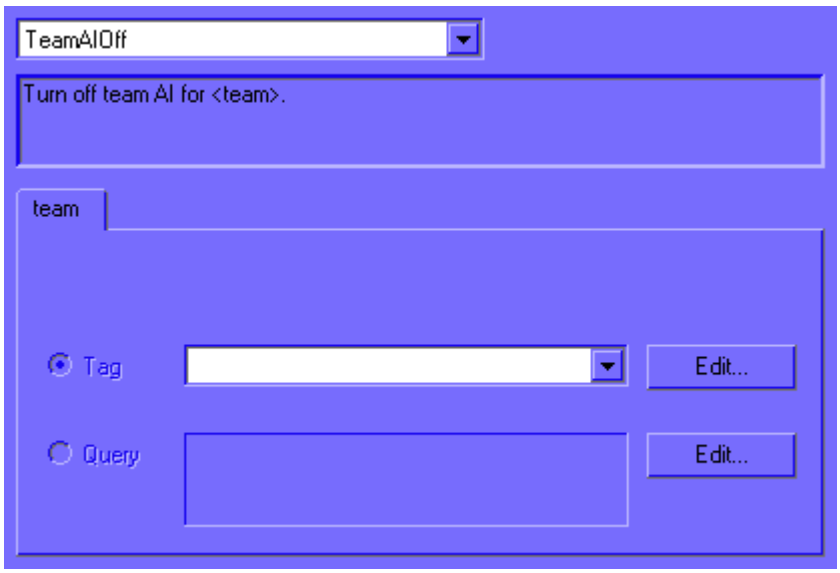
This Response changes the maximum AI spotting distance to the designated value, effectively overriding the environment file settings. Its only Parameter is Distance.

StopIf



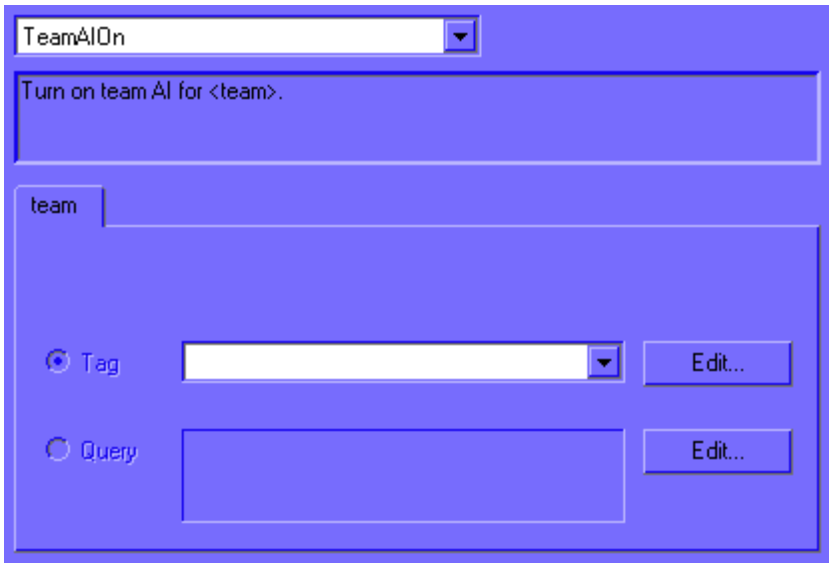
This Response determines if the Responses below it will be activated or not depending on certain conditions. If the conditions are true, the rest of the Responses will not be activated. It's only Parameter is Condition.

TeamAIOff



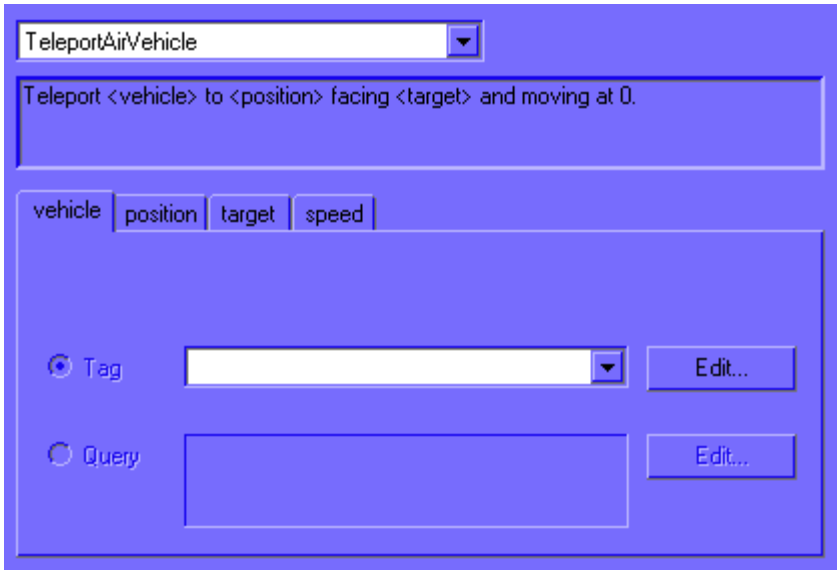
This Response turns off the Team-level AI for the designated Team. Its only Parameter is Team.

TeamAIOn



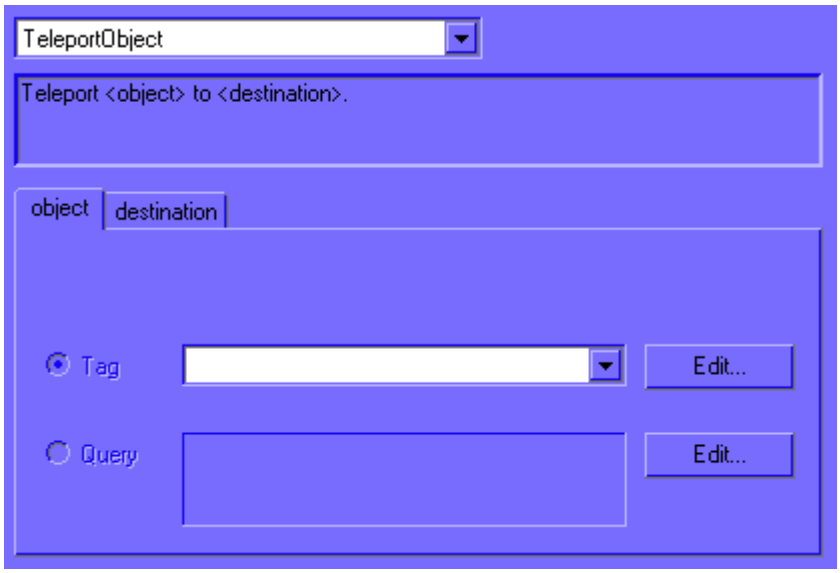
This Response will turn on the Team-level AI for the designated Team if it has been shut off using the TeamAIOff Response. Its only Parameter is Team.

TeleportAirVehicle



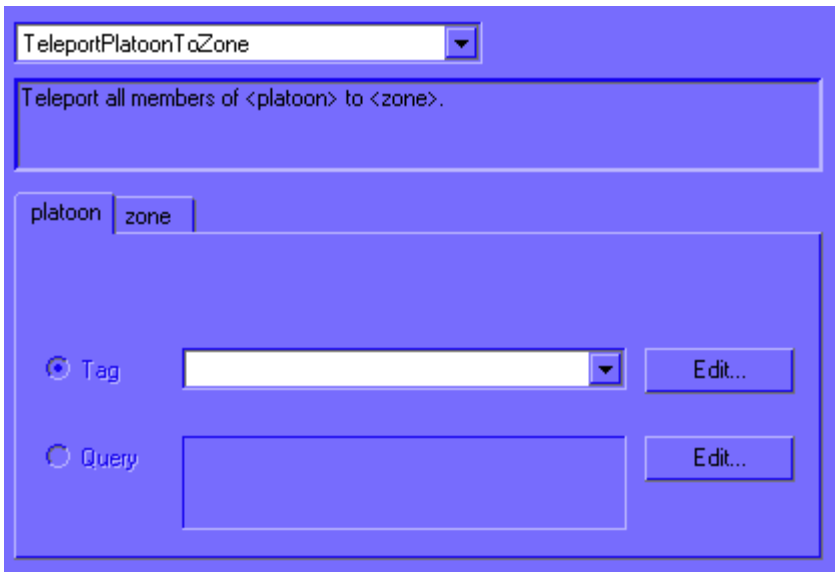
This Response teleports the designated Vehicle (which must be a helicopter) to the specified location, facing the specified target, and moving at the designated speed. It should only be used with helicopters, and altering the speed may cause unexpected vehicle behaviors. Its Parameters are Vehicle, Position, Target, and Speed.

TeleportObject



This Response teleports the designated object to the specified destination. It should only be used with Actors and Vehicles, and it should not be used to teleport them from one Map Room to another. Its Parameters are Object and Destination.

TeleportPlatoontoZone

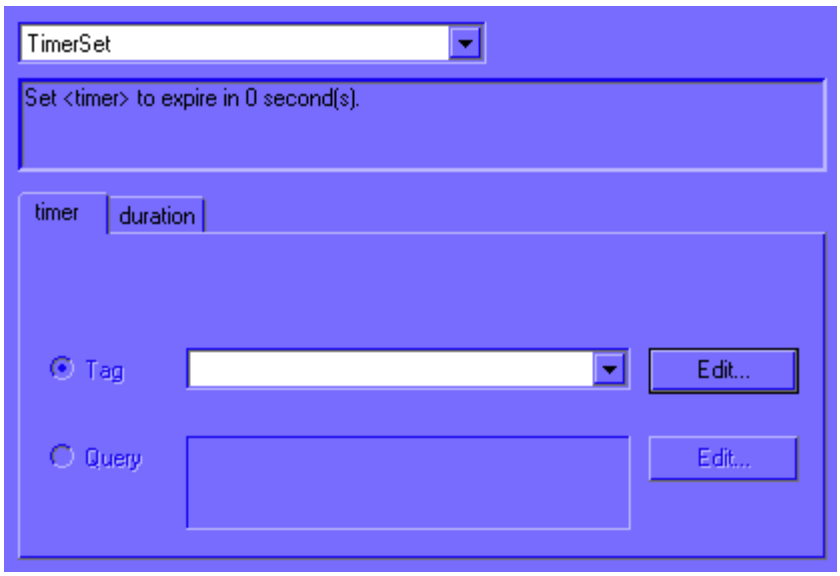


This Response teleports the designated Platoon to the specified Zone. It should not be used to teleport them from one Map Room to another. Its Parameters are Platoon and Zone.

TimerKill

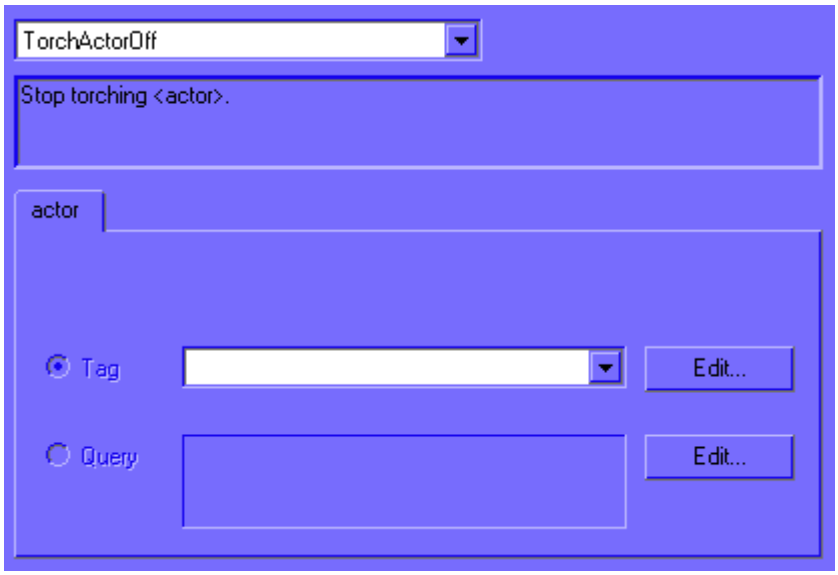
This Response stops the designated timer without triggering any blocks triggered off of that timer expiring. Its only Parameter is Timer.

TimerSet



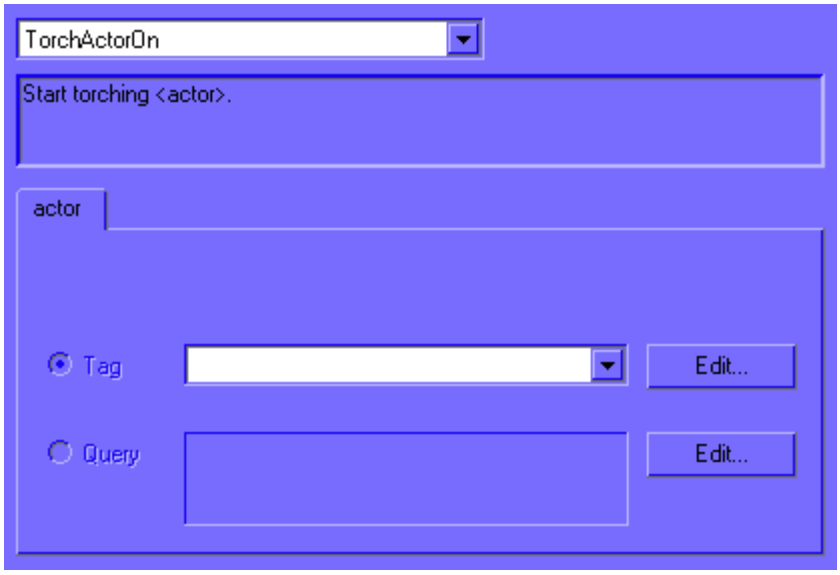
This Response sets the designated timer to expire in the specified amount of time. Its Parameters are Timer and Duration.

TorchActorOff



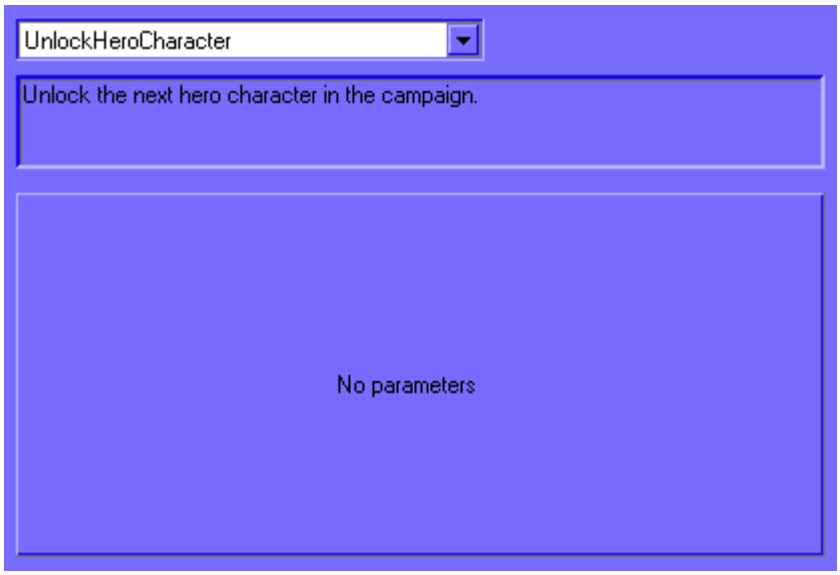
This Response stops the flames at the designated Actor's feet that started with the TorchActorOn Response. Its only Parameter is Actor

TorchActorOn



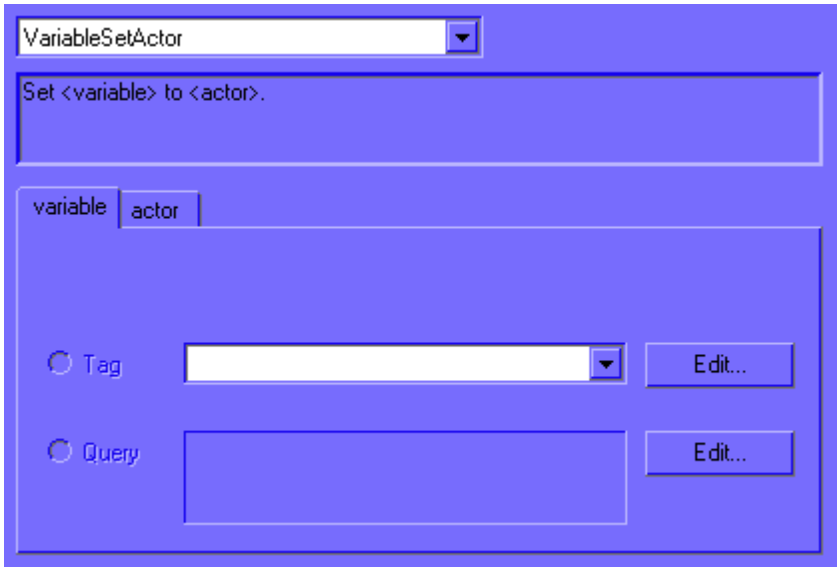
This Response makes flames appear at the feet of the designated Actor, as in the Cat and Mouse multiplayer game type. Note that this flame is not damaging to an Actor. The only Parameter is Actor.

UnlockHeroCharacter



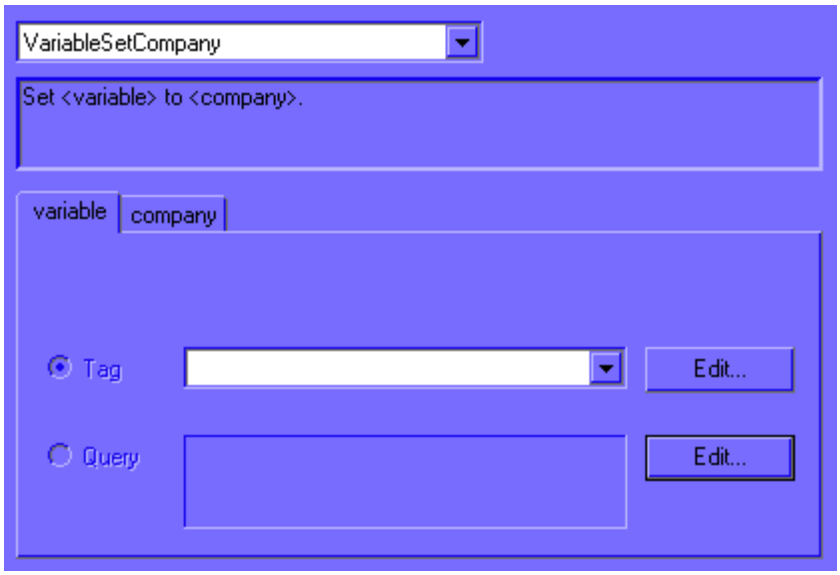
This Response unlocks the next hero character in the campaign as specified in the campaign file (see “Campaign File Editor” in the Tools section for more information) once the mission ends successfully. It has no Parameters.

VariableSetActor



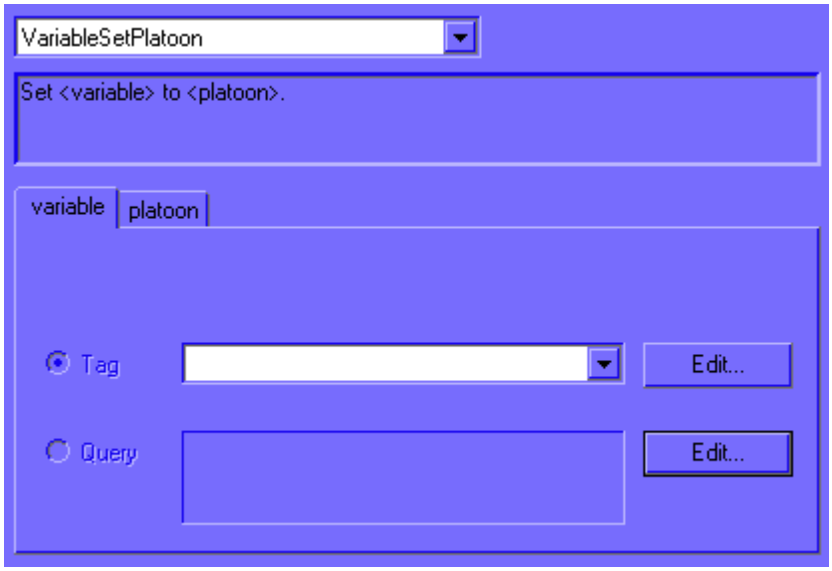
This Response sets a Variable to reference the designated Actor. Its Parameters are Variable and Actor.

VariableSetCompany



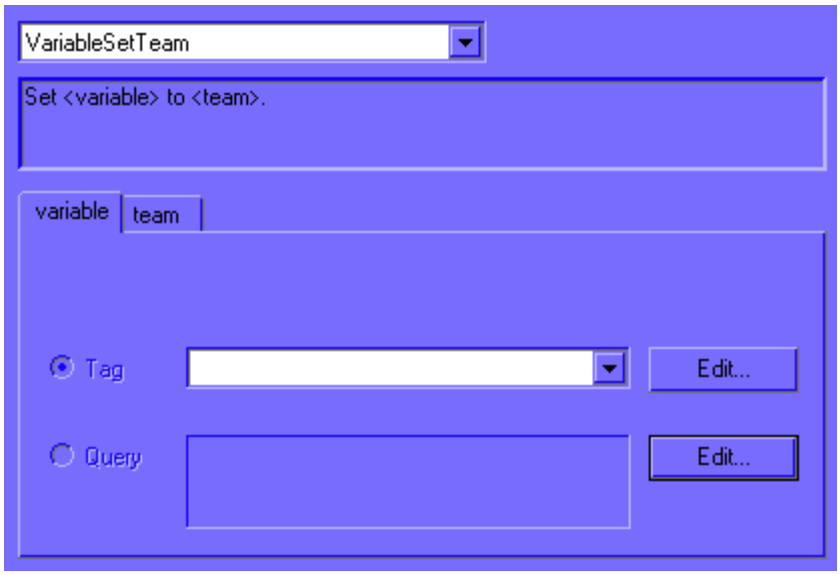
This Response sets a Variable to reference the designated Company. Its Parameters are Variable and Company.

VariableSetPlatoon



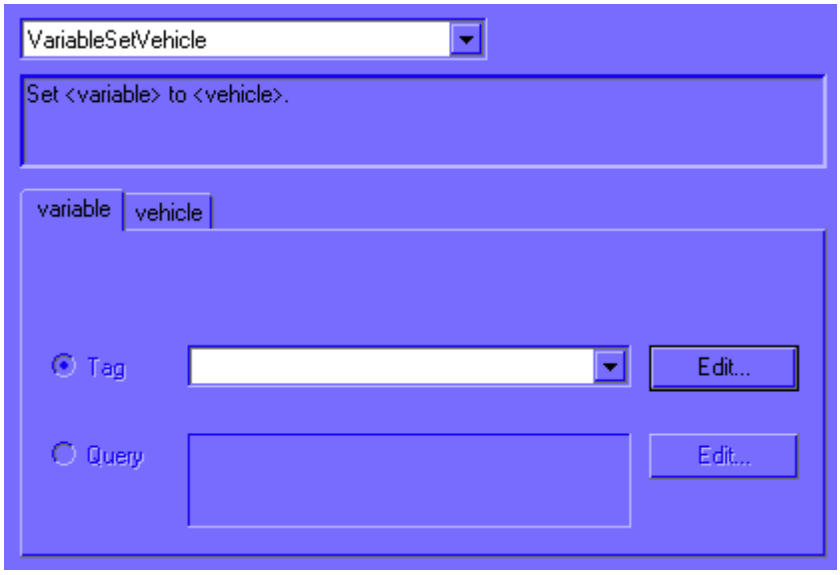
This Response sets a Variable to reference the designated Platoon. Its Parameters are Variable and Platoon.

VariableSetTeam



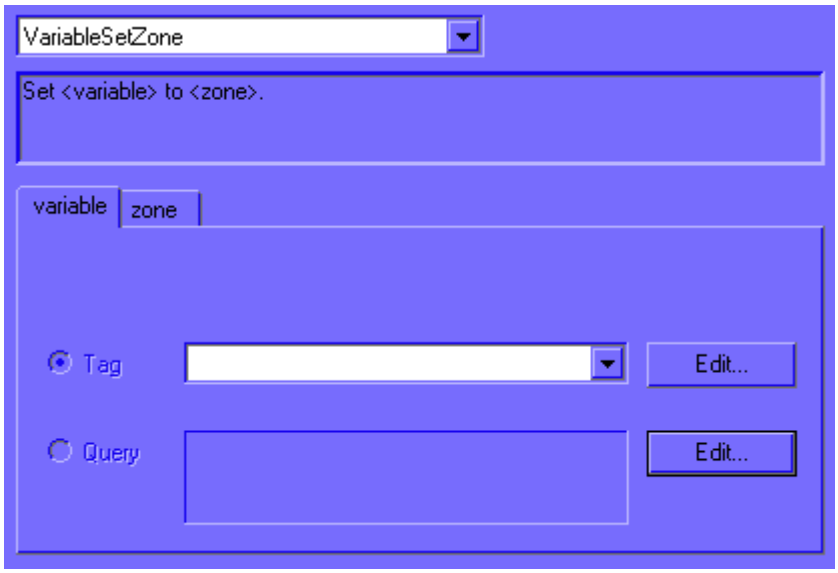
This Response sets a Variable to reference the designated Team. Its Parameters are Variable and Team.

VariableSetVehicle



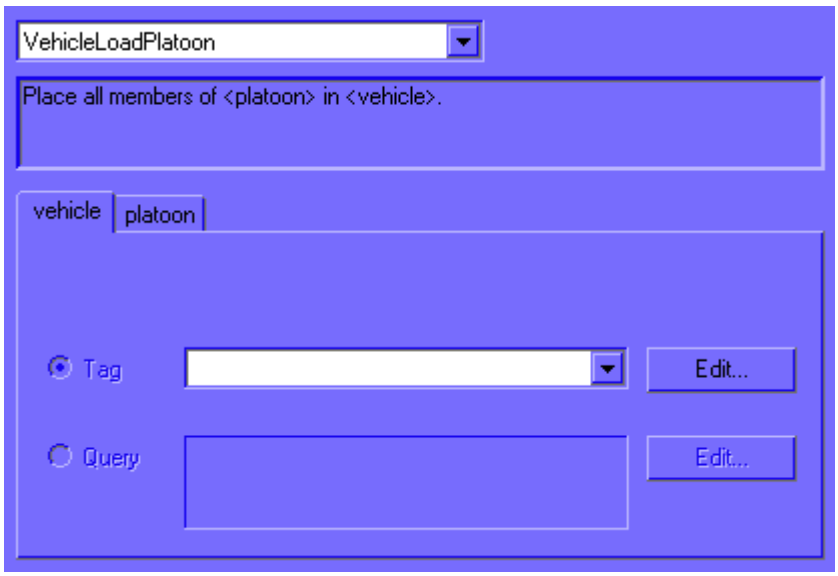
This Response sets a Variable to reference the designated Vehicle. Its Parameters are Variable and Vehicle.

VariableSetZone



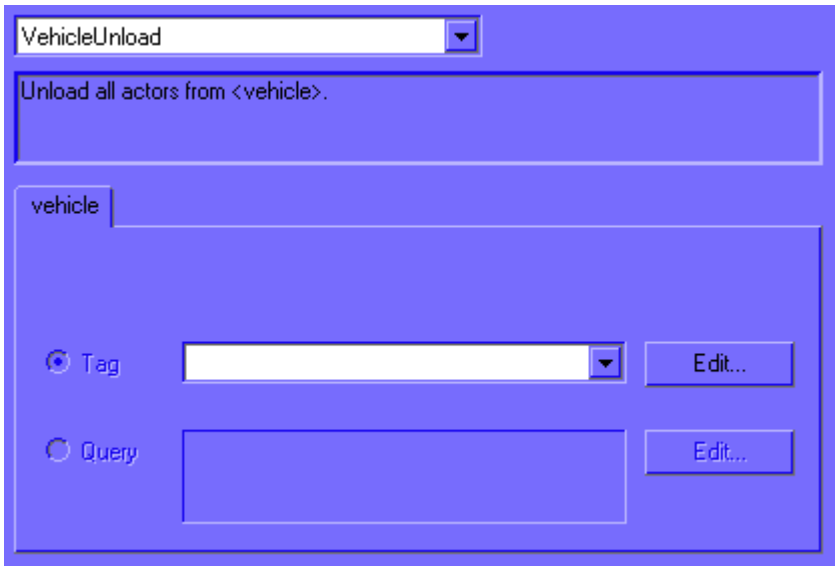
This Response sets a Variable to reference the designated Zone. Its Parameters are Variable and Zone.

VehicleLoadPlatoon



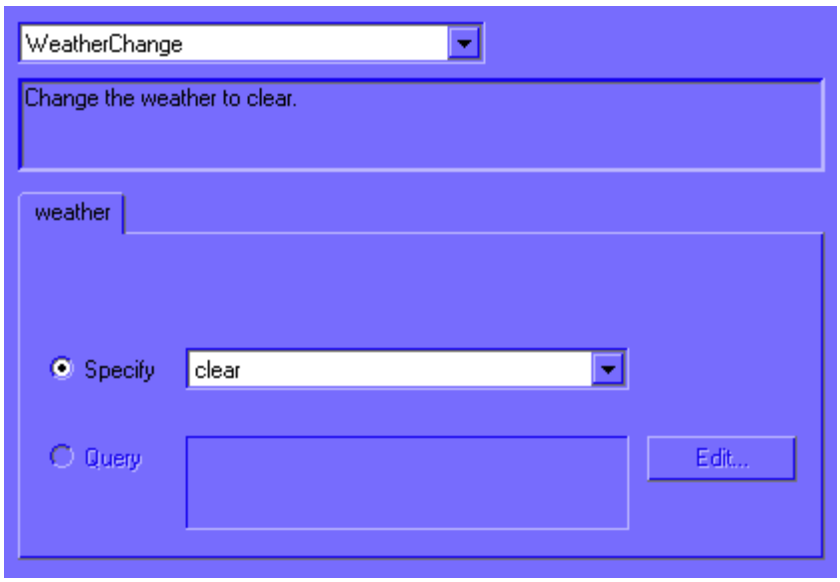
This Response places all members of the designated Platoon into the specified Vehicle. Its Parameters are Vehicle and Platoon.

VehicleUnload



This Response unloads all Actors from the designated Vehicle. Its only Parameter is Vehicle.

WeatherChange



This Response changes the mission's weather to the designated weather type. The only Parameter is Weather.

Parameters

Parameters are the different options available to use in various Triggers and Responses. Each Parameter covers a different kind of Element, and gives you different ways of identifying that Element, using Identifiers. Most Identifiers allow you to use a pull-down menu that has a list of Elements that fall under the Parameter's category. Others allow you to type in a specific Element or a file name, or to choose a constant. The most powerful Identifier is a Query – it lets Igor figure out what should be identified, and is thus more flexible than other Identifiers.

Parameters are separated into categories based on their Identifiers, since Identifiers determine how the Parameter is used. For example, since Actor, Team, Company, and Platoon all have the same types of Identifiers, they're in the same category. What effect each Identifier has on a Block is shown in the summary section of the window.

Note that no Trigger Parameters have the Query Identifier, even if that same Parameter includes the Query Identifier in a Response.

Common Element Parameters

Most Identifiers that identify Elements are used the same way. They include the Tag and Query Identifiers.

Tag is the pull-down menu that lists Placed Elements or blank Element References. Clicking "Edit…" next to Tag will bring up the Tags dialog, where the user can create a blank Element Reference of the appropriate type. The Tag Identifier will only display Elements in the pull-down menu of the appropriate Element type for that Parameter. For example, the Platoon Parameter will only show Platoons in the Tag pull-down menu.

The Parameters that fall into this category are Actor, Company, Counter, Destination, Effect, Flag, Group, Location, New, Objectives, Old, Plan, Platoon, Player, Position, Target, Team, Thing, Timer, Variable, Vehicle, Victors, and Zone.

Number Parameters

Any Parameters that reference a number value falls into the Number Parameter category. Number Parameters include the Literal, Constant, and Query Identifiers.

Literal is a numerical value that the user enters. These can include decimals. Constants are Element References that have a value assigned to them elsewhere in the script.

The Parameters that fall into this category are Blue, Distance, Duration, Far, Green, Loop, Near, Range, Red, Score, Size, Speed, Time, Track, and Volume.

File Parameters

Parameters that reference an outside file, such as a wav or .avi file, are part of the File Parameters category. They include the Literal, Constant, and Query Identifiers.

Literals in this kind of Parameter are not numerical values, but are instead filenames and extensions. The user types the filename and extension into the Literal section exactly as the filename appears to the operating system. For example, to use PlaySound to play a sound called "a_exmetalwav", the user would type a_exmetalwav into the Literal Identifier. Likewise, a Constant can be created here and assigned a file elsewhere in the script.

The File Parameters are Female, File, Kit, Male1, Male2, and Sound.

Text Parameters

Parameters that display messages in some form or another are Text Parameters. Text Parameters include the Literal, Constant, and Query Identifiers.

Literals in this kind of Parameter are where the user types in the message to be displayed. Constants are messages defined elsewhere in the script.

The Text Parameters are Losetext, Message, Text, and Wintext.

Manual Element Parameters

Manual Element Parameters are like Common Element Parameters, except that instead of being able to select an Element through the Tag Identifier, the user must type in the Element in the Literal Identifier. Manual Element Parameters include the Literal, Constant, and Query Identifiers.

For Manual Element Parameters, the Literal Identifier is where the user manually enters the name of the Element that needs to be identified. For example, if the user wanted to open a door with the OpenDoor Response, he'd type in the name of the door as a map object into the Literal section of the Door Identifier.

The Manual Element Parameters are Command, Object, and Door. Command is the name of a console command.

Misc Parameters

Some Parameters fall into their own categories. These are lumped into the Misc Parameters Category by default. These Parameters and their specific Identifiers are detailed below.

Weather - Weather uses the 'Specify' Identifier. The user can choose between clear, rain, and snow.

Condition - Condition uses the Specify and Query Identifier. Specify can be either true or false, but Query is used more often here.

Highlight - Highlight can be either true or false, and specifies if a Zone will emit smoke.

Color - Color can be neutral, red, blue, green, or gold – colors which correspond with multiplayer Platoon colors.

State - State is either true or false, and usually identifies the state of a Flag.

Decoration - Decoration identifies the kind of decoration used in Responses that give out decorations.

Queries

As discussed previously, Queries are Identifiers, but slightly different than other Identifiers. In a Query, the user doesn't specify something; instead, they basically ask Igor a question about something that needs identified, and Igor answers the question while the game is running.

For example, ContinueIf and StopIf basically do nothing if they don't use Queries. You'll see summaries like "Continue executing responses if True". However, when used with a Query that can see if the number of Actors on the player's Platoon are equal to the number of Actors in that Platoon within a certain Zone, it suddenly becomes the one thing that defines an Extraction Zone.

What follows is a list of all the possible Queries taken directly from the Igorscripting.txt that is in the Ghost Recon base directory, reprinted here for your convenience with special thanks to David Hamm. Note that not every Parameter that includes "Query" as an Identifier will use all of these.

Query Reference

--Flags and counters—

GetFlagState
"The state of <flag>"
Used to retrieve the state of a previously set flag. If <flag> has never been explicitly set, GetFlagState will return false.

GetCounter
"The value of <counter>"
Used to retrieve the value of a previously set counter. If <counter> has never been explicitly set, GetCounter will return 0.

--Integer arithmetic—

GetSum
"The result of <value1> plus <value2>"
GetDifference
"The result of <value1> minus <value2>"
GetProduct
"The result of <value1> multiplied by <value2>"
GetQuotient
"The result of <value1> divided by <value2>"
Simple integer arithmetic. If division by zero is attempted, the dividend will be returned.

--Floating point arithmetic--

GetSum
"The result of <value1> plus <value2>"
GetDifference
"The result of <value1> minus <value2>"
GetProduct
"The result of <value1> multiplied by <value2>"
GetQuotient
"The result of <value1> divided by <value2>"

Simple floating point arithmetic. If division by zero is attempted, the dividend will be returned.

--Type conversions--

ConvertIntegerToNumber
"The number <value>"
ConvertNumberToInteger

"<value> truncated to an integer"

--Equalities--

CompareIntegers
"<value1> <equality> <value2>"
CompareFloats
"<value1> <equality> <value2>"
CompareThings
"<thing1> <equality> <thing2>"

Comparison operations. Note that floats cannot be compared for equality.

--Logical operations--

And
"<value1> and <value2>"
Or
"<value1> or <value2>"
Not
"The inverse of <value>"

Logical operations performed on boolean values.

--Statistics--

GetKillsActor
"The number of kills for <actor>"
GetKillsCompany
"The number of kills for <company>"

These queries provide access to the running statistics tracked for players and companies.

GetActorMostKills
"The actor with the most kills"
GetCompanyMostKills
"The company with the most kills"

If there is a tie, these references will be invalid. Use ActorValid to check.

--Campaign difficulty settings--

DifficultyEasy
"The campaign difficulty is set to Easy"
DifficultyNormal
"The campaign difficulty is set to Normal"

DifficultyHard
"The campaign difficulty is set to Hard"

These tests will reveal the campaign difficulty setting.

--Random number generation--

RandomInteger
"A random integer between <low> and <high>"
RandomNumber
"A random number between <low> and <high>"

These queries generate random values between (and including) the given bounds.

--Active objects--

ActorActive
"<Actor> is active"
GetActiveActors
"The number of active actors"
GetFirstActiveActor
"The first active actor"
GetActiveCompanies
"The number of active companies"
GetFirstActiveCompany
"The first active company"
GetCurrentTeamSize
"The number of active actors on <team>"
GetCurrentPlatoonSize
"The number of active actors on <platoon>"
GetCurrentCompanySize
"The number of active actors on <company>"

These queries get information on which actors are not out of action.

--Parent access--

GetActorTeam
"The team including <actor>"
GetActorPlatoon
"The platoon including <actor>"
GetActorCompany
"The company including <actor>"
GetVehicleCompany
"The company including <vehicle>"

These queries deal with the unit hierarchy.

--Shooter access--

GetActorShooter
"The shooter who killed <target>"
GetVehicleShooter
"The shooter who destroyed <target>"

Use ActorValid to check the reference returned by these queries. There may be cases where a kill is not attributed to a specific actor.

--Reference validation--

ActorValid
"<Actor> is a valid reference"
TeamValid
"<Team> is a valid reference"
PlatoonValid
"<Platoon> is a valid reference"
CompanyValid
"<Company> is a valid reference"
VehicleValid
"<Vehicle> is a valid reference"
ZoneValid
"<Zone> is a valid reference"

Several queries return references. Use this query to see if the reference is meaningful. The responses will return false if the reference has not been initialized.

--Player access--

GetPlayerActor
"The player-controlled actor"

This query is intended for single-player use.

GetPlayerPlatoon
"The player-controlled platoon"
GetPlayerCompany
"The player-friendly company"

These queries are intended for coop or single-player use.

--Hidden objects--

IsHidden
"<Thing> is currently hidden"
IsShown
"<Thing> is currently shown"

Specify a company, platoon, team, actor, or vehicle. All members much match the criteria or the query will return false.

--*Room lighting*--

LightsOn
"The lights are on in <room>"
LightsOff
"The lights are off in <room>"
GetRoom
"The room containing <location>"

--*Map object states*--

MapObjectDestroyed
"<Object> has been destroyed"
DoorOpen
"<Door> is open"
DoorClosed
"<Door> is closed"

Use to query the state of objects in the map.

--*Escort state*--

IsBeingEscorted
"<Team> is currently being escorted"
GetEscortCompany
"The company currently escorting <team>"

Queries for information about captives and hostages.

--*Detection*--

ActorSeenByCompany
"<Actor> is being seen by <observers>"
ActorHeardByCompany
"<Actor> is being heard by <listeners>"
TeamSeenByCompany
"A member of <team> is being seen by <observers>"
TeamHeardByCompany

"A member of <team> is being heard by <listeners>"
PlatoonSeenByCompany
"A member of <platoon> is being seen by <observers>"
PlatoonHeardByCompany
"A member of <platoon> is being heard by <listeners>"
CompanySeenByCompany
"A member of <company> is being seen by <observers>"
CompanyHeardByCompany
"A member of <company> is being heard by <listeners>"
ActorSeenByActor
"<Actor> is being seen by <observer>"
ActorHeardByActor
"<Actor> is being heard by <listener>"
PlatoonSeenByPlatoon
"A member of <platoon> is being seen by <observers>"
PlatoonHeardByPlatoon
"A member of <platoon> is being heard by <listeners>"

Vehicles are not considered in these queries, neither as a target nor source for detection. An actor counts as "heard" if any sounds generated by the actor (including gunshots) have been detected in the past two seconds.

--*String operations*--

ConvertIntegerToString
"<Value> as text"
ConvertNumberToString
"<Value> as text"
GetActorOwnerName
"The owner of <Actor>"
ConcatenateStrings
"<Start> + <end>"

These queries can be used to construct messages for display.

--*Proximity*--

TeamMembersAtLocation
"The number of members of <team> within <range> meter(s) of <location>."
PlatoonMembersAtLocation
"The number of members of <platoon> within <range> meter(s) of <location>."
CompanyMembersAtLocation
"The number of members of <company> within <range> meter(s) of <location>."

Count actors at a location using the same rules as the proximity triggers.

--Platoon capabilities--

PlatoonDemoChargeCount
"The number of demo charges available to <platoon>"
PlatoonAntiTankRocketCount
"The number of anti-tank rockets available to <platoon>"

These queries can be used to determine whether a platoon has the components required to complete certain mission objectives. Only living members of the platoon are considered.

--Zone access--

GetCentralArea
"The central area"
GetReconInsertion
"The recon insertion zone"
GetReconExtraction
"The recon extraction zone"
GetBase
"Base <index>"

Used by game type scripts to reference map zones. If using these queries in a mission script, be sure the requested zones exist.

--Spotlights--

SpotlightHasTarget
"<Spotlight> has a target"
SpotlightGetTarget
"The target of <spotlight>"

Query the state of spotlights, which are map objects.

--Game mode--

ModeSP
"The game mode is Single Player"
ModeCoop
"The game mode is Coop"
ModeSolo
"The game mode is Solo"
ModeTeam
"The game mode is Team"

Used to determine the current game mode. Some scripting commands are only safe in single player games.

--Miscellaneous--

ActorWounded
"<Actor> is wounded"

A wounded actor may or may not be out of action. Combine this query with ActorActive for more information.

GetRange
"The distance between <location1> and <location2>"

Returns the 3D distance between two objects in meters. If either object is unknown the distance is considered 1 billion meters.

GetNearestActor
"The actor nearest <location>"

If <location> is an actor, it will not be considered in the check.

GetCompanyScore
"The score for <company>"

Query a company score counter.

CompanyPresent
"A player on <company> is connected"

Used to detect when players leave a game before the after action.

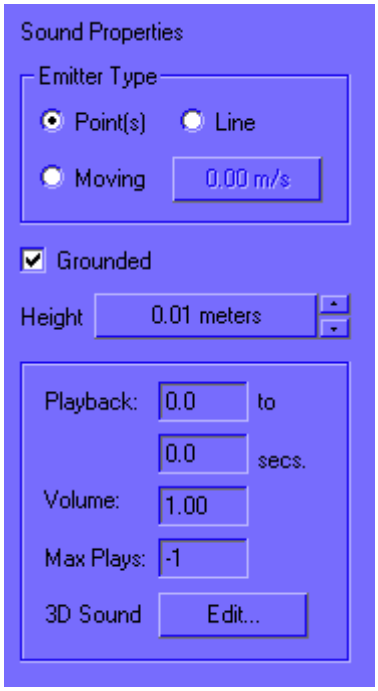
GetCompanyColor
"The color of <company>"

Get the color associated with the given company. Used with the MapZoneColor response.

Sounds

Sound is one of the most immersive aspects of Ghost Recon. A scene of a beautiful forest swaying in the wind is one thing, but to hear the wind rustling through the leaves and hearing the chirping of birds when a gunshot rings out in the distance adds that final layer of reality to the simulation.

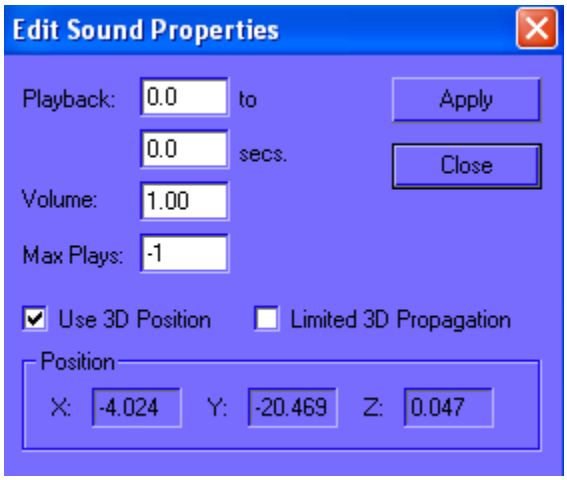
A Sound is placed like most other Elements – select a Sound from the Sound tab in the Available Elements List and then ctrl click in the desired location for the Sound in the 2d Command Map window. After that, the Sound Properties window is used to make the Sound seem real.



Sounds emit the specified wav file in different ways depending on what kind of Emitter Type the user has selected. A Point Emitter emits sound from the exact point the user has placed the Sound. Line Emitters will emit the same sound from each point along the line, which is placed much like the Path Plan Step. Moving Emitters will emit sound from a moving point at the designated speed along a specified path.

If a Sound is grounded, its height is determined by its distance in meters away from the ground. If it is not grounded, its height is determined by its place along the Z axis of the map. It is recommended that Sounds are grounded.

The final section of the Sound Properties window determines how many times and how often a Sound will play. By clicking the Edit button, the Edit Sound Properties window will appear.



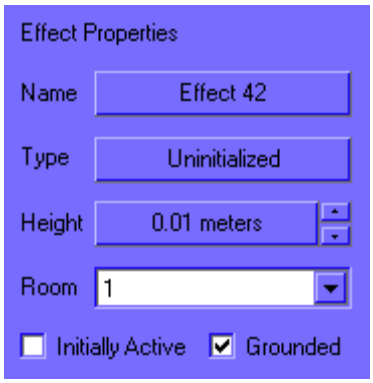
The Playback section marks an interval of time in which the Sound will play. For example, if a Sound is supposed to play between every five to thirty seconds, 5 is entered as the first number and 30 is entered as the second number. The sound will then play at a random number of seconds between 5 and 30.

A Volume of 1 is max volume for the given Sound. Reducing this number reduces the volume of the Sound. If a Sound is supposed to start at a 3d source (which makes it fade out at a distance, become louder as a player approaches it, and use any environmental audio effects), "Use 3d Position" must be checked. "Limited 3d Propagation" should only be used for Sounds where exact 3d details don't need to be used, like thunder. Click "Apply" to save any changes made before closing the Edit Sound Properties window.

Effects

When you see a scripted explosion or a camera switch during a mission, you're seeing the results of scripted Effects. Effects are an Element that has a wide variety of uses, from the Effect being used to make Hollywood-like special effects, to being a placeholder location that you can reference in the Scripting Language.

An Effect is placed like most other Placed Elements. Ctrl-clicking on the 2d Command Map after "Effect" is chosen from the Misc tab of the Available Elements List. Usually, the direction an Effect is facing doesn't matter.



Once an Effect is placed, its Properties can be changed in the Element Properties window. Type refers to the specific kind of special effect the Effect will produce if the ActivateEffect Response is used in the Scripting Language. The Height refers to either the number of meters away from the ground the Effect is at if "Grounded" is checked, or the Z value of the Effect in relation to the map coordinates if "Grounded" is not checked. Room refers to the Map Room the Effect is supposed to be part of – it will default to the Room it is placed in and it is not recommended that this value be changed. If "Initially Active" is checked, it is the same as using the EffectActivate Response in a block triggered by Startup.

Effect List

This is a list of the different special effects that can be created by altering an Effect's Type.

Common Effects

These Effects are commonly used in Ghost Recon. In fact, the different types of Effects listed after these are largely untested, so use them at your own risk. What's listed here must be typed verbatim into the Type section of the Effect Properties. The parameters after that, indicated by parentheses, are modifiable.

General_type14(delay)(size) - This is the explosion used in M08 from GR for the napalm strike, in D05 from Desert Siege when the player blows up the plane parts, in C04 in Island Thunder when the ammunition building explodes, and in C08 from Island Thunder when various buildings explode in the main enemy base. The first parameter is the number of seconds the explosion will be delayed when it is activated. The second parameter is the size of the explosion. Note that this is a 2d effect that will always face the player.

General_type41(?)(?)(?)(delay) - This is the debris used when the above explosion is used. The first three parameters don't do anything, but the fourth one sets the delay. The first three should be set at 0, and those three must be in place before the delay can be set.

Damage(range)(?) - This is the damage that the above explosion is supposed to cause. The first parameter is the range in increments much larger than meters, and the second is the delay. The range shouldn't be greater than 2 or nothing will happen.

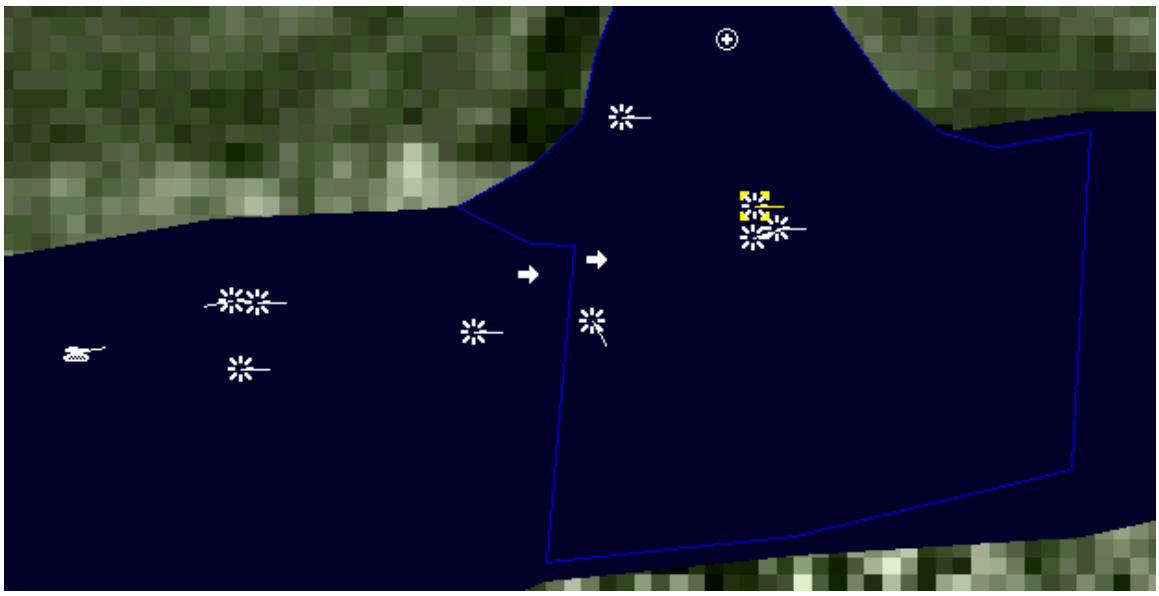
The other effects are in need of research and are not detailed here yet

Effects can also be used as positions for Cameras and Sounds that are to be referenced in the Scripting Language. The PlaySound3d or CameraSwitch Responses can both reference placed Effects. See these Responses in the Response list above for more information.

Camera Scripting

Different Camera views in Ghost Recon are easy to use but extremely limited in their functionality. Cameras in GR are fixed – there is no pan, no zoom, no tracking, etc. They are stationary and point to a stationary target. The camera can be told to switch to different Elements and point to different Elements using the CameraSwitch Response. For total control over a camera shot, it is recommended that an appropriately named Effect be used for the camera and another for the camera target. For example, an Effect named Camera Position A and an Effect named Camera Target A are easy to identify in the Scripting Language for use with CameraSwitch.

Before switching the camera out of the default 1st person view, make sure that the CinematicModeOn Response is used, or the player may retain control of his character or other anomalies may occur.

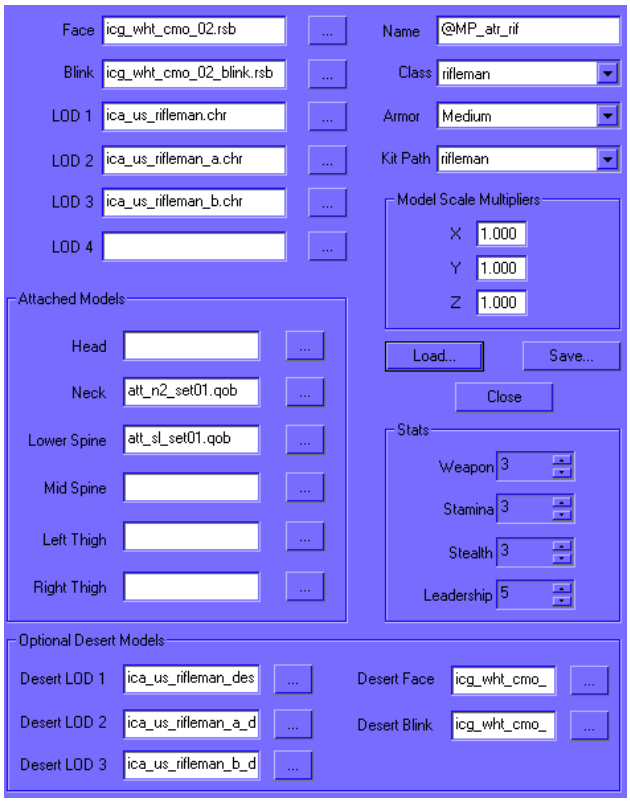


Multiple cameras and positions can be used with some good timing to create cinematic events. The vehicle insertions seen in Island Thunder are mainly comprised of blocks that control the timing of the camera switches, with the other blocks ensuring that the cinematic can be skipped by user input and that other strange occurrences don't happen because of that skipping. The picture above is the camera placement for the insertion cinema for C05 of Island Thunder. The specifics of vehicle insertion cinemas will be covered in the Appendix.

Single-Player Tools

Igor has a variety of tools that can change a mission more indirectly than scripting. For single player, the tools that will concern you the most are the Actor, Campaign File, Combat Model, Environment, Gun, Hand Held Item, Kit, Projectile, Sound Volume, and Vehicle Editors. All these Editors can be used to create or modify files outside the mission file to change the way things look or act within the mission.

Actor Editor



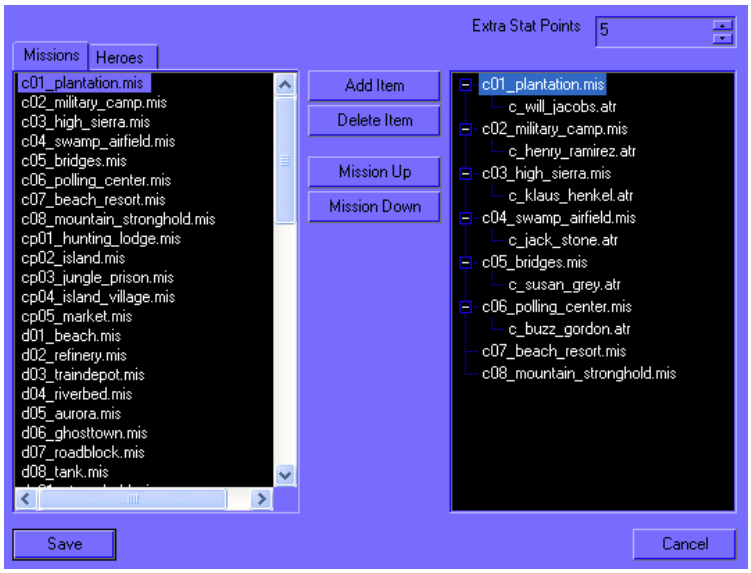
The Actor Editor can be used to change or create Actor Files. The user can enter the rsb's for an Actor's face and blink face, the chr file for an Actor's model and model LoD's, the Actor's name (that appears when IFF is turned on or when they kill someone that causes a message to be displayed in multiplayer games), class, armor level, etc.

The file name can either be typed in verbatim, or it can be selected through by browsing directories by clicking on the "…" buttons.

Note that if an Actor's scale is changed, all three scale values must be the same or else their distortions will cause graphical anomalies.

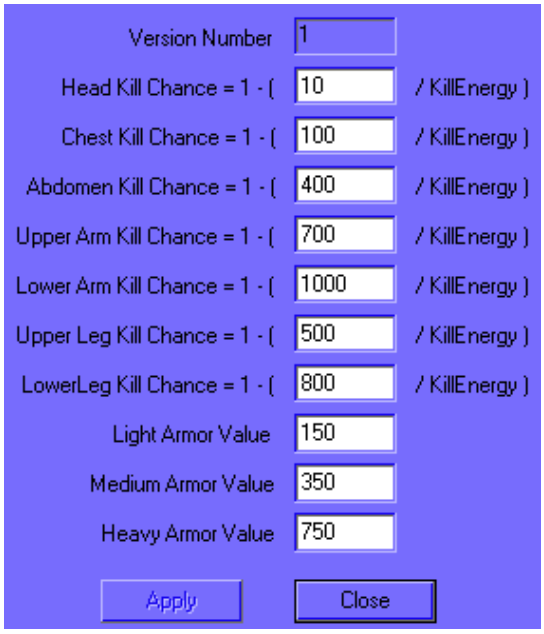
Actor files can also be edited by opening the atr file with a text editor, but that method is error-prone. It's always best to use the editor in Igor.

Campaign File Editor



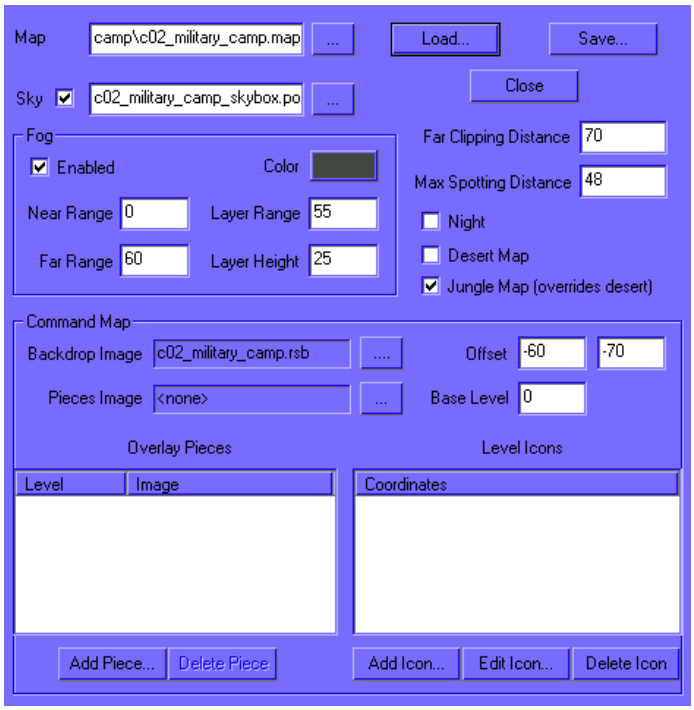
The Campaign File Editor lets the user quickly and easily set up a series of missions, their order, and the heroes unlocked after each mission. The Campaign File Editor will edit the campaign file for the current active mod. On one side it will list all of the available missions under one tab and all the available heroes under the other. On the other side is the current mission order and hero unlocking order. To add a mission to the campaign, simply select it on the left and then click "Add Item". To add a hero to be unlocked, click on the added mission on the right side, highlight the hero on the left and click "Add Item". Once missions are placed, clicking "Mission Up" or "Mission Down" will move their order in the campaign. Click Save when done and it will automatically save to the correct campaign file for the active mod. Note that you can also set the number of combat points available to the player's forces at the beginning of the campaign by changing the value in the upper right-hand corner, labeled "Extra Stat Points".

Combat Model Editor



The Combat Model Editor lets the user change the way damage is handled in GR. Basically, the lower the number/the Kill Energy is, the greater the chance that a hit from there will kill. For armor values, higher numbers prevent more damage. The Kill Energy is derived from the gun file, which can be changed using the Gun Editor.

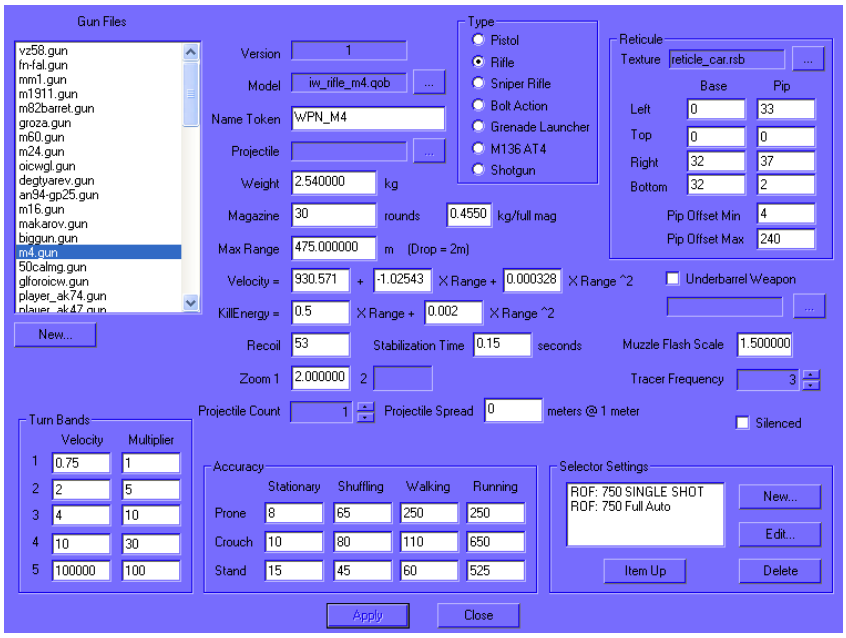
Environment File Editor



The Environment File Editor lets the user adjust the sky box, fog plane and color, clipping distance, time of day, type of map, command map picture and centering, and icons indicating different levels on the command map. Although many of these values can be overridden in the script or through the Map Properties, in the end the values should be determined by the environment file.

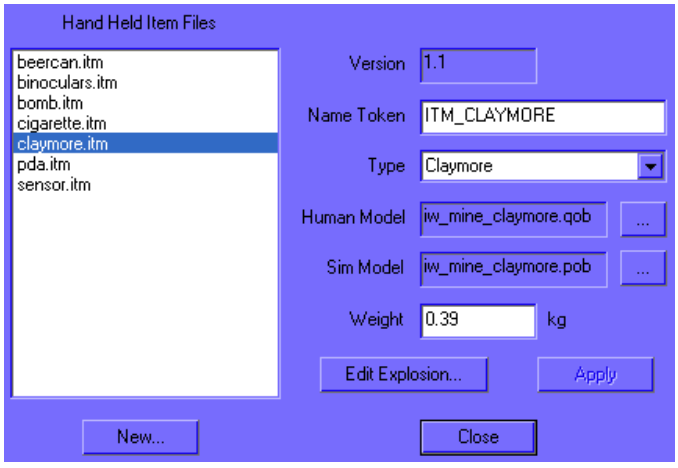
Each map file can have multiple environment files. This can be useful if the user is making a mod and wants to change the fog and skybox of a given map – the user doesn't have to change the existing env file, he can just make a new one. All distance values in the Environment File Editor are in meters. To enter the Level Icons, look in the lower left-hand corner of the Igor screen, and move the mouse over an area in the 2d Command Map window. Note that x/y/z coordinates are given. If there's a map with a staircase or other structure leading to the next planning level, find these coordinates and enter them in the Environment File Editor. They will appear on the command map as numbers, and pressing C or X with the command map open in GR will cycle through the available planning levels. Each level needs a new icon – that is, if there is an icon for the first floor that says "1", and icon saying "2" will have to be created, and "3" and so on, for each planning level. These are optional, and only for user reference and information. Do not use the "Overlay Pieces" section, as it is not fully implemented.

Gun Editor



The Gun Editor lets the user edit every attribute of all of the firearms in GR, and make new guns. The best way to learn the Gun Editor is to browse existing guns after becoming familiar with their operation in GR and see what results occur when those gun files are modified.

Handheld Item Editor

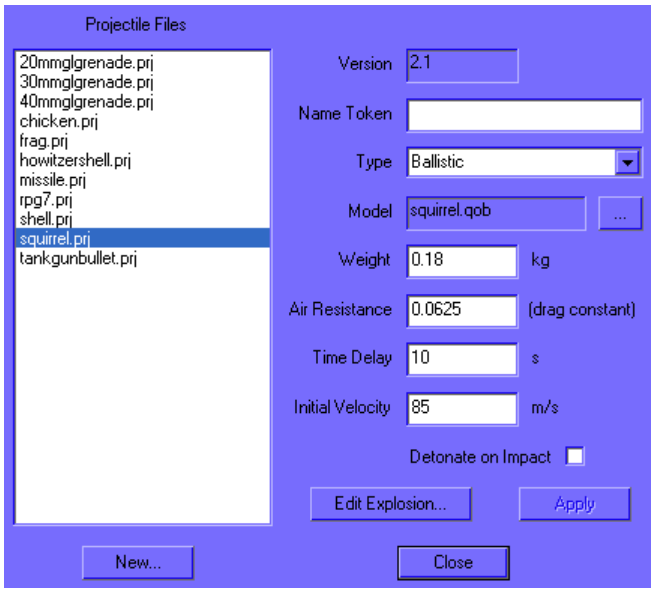


The Handheld Item Editor lets the user edit the non-firearm items that the Ghosts can equip, such as sensors and claymores. Values such as the item type, the model it uses, and its weight can be altered here. After changes have been made, click Apply to save them.

Kit Editor

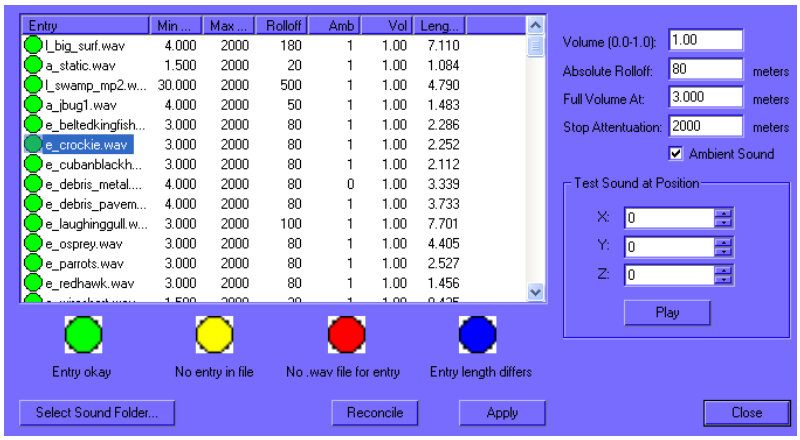
The Kit Editor lets the user modify or create new kit files. Kits are the files that determine what weapons any given Actor is carrying. To create a new kit or alter a kit by selecting a new weapon, click "Browse…" under the Slot 1 or Slot 2 section, and choose the appropriate prj or gun file. The Mag Count and Count sections are how much ammo the Actor will carry for the weapon. Click "Save" when done to save a new kit file.

Projectile Editor



The Projectile Editor lets the user change the attributes of a given projectile or create a new projectile. Most projectiles are fired from other weapons but some, such as the frag.prj, are handheld items. Type can be Ballistic (affected by gravity) or Dumb Missile (not affected by gravity). The model can be any pob or qob file, Time Delay is how long of a pause there is before the projectile can be used again, and Initial Velocity is how fast it's moving when it is used. Clicking "Apply" will save all changes or it will save the new projectile.

Sound Volume Editor

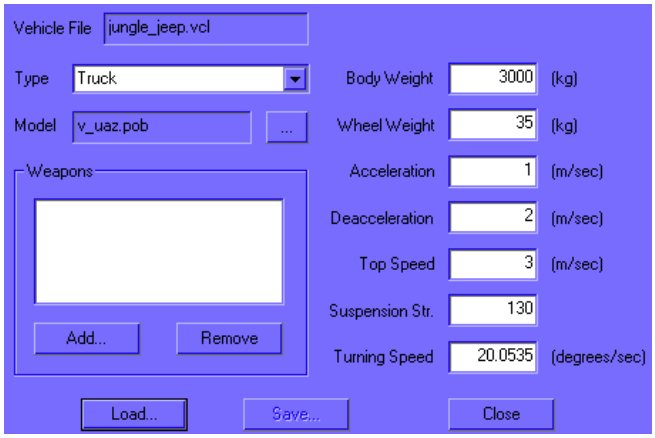


The Sound Volume Editor lets the user alter the properties of existing sounds or put new sounds into GR. Any wav files in the sound directory of the active GR mode will appear in this list. To change what directory the Sound Volume Editor is looking at, click on "Select Sound Folder". Any new wav file placed into any sound directory should be 22khz. Once it is placed in the sound directory, it will appear in the list in the Sound Volume Editor with a yellow circle next to it. Sounds with anything other than a green circle cannot be used by GR yet. First, the user has to click "Reconcile" and then "Apply" – this should make the circles green. If a circle is red, the sound was added to the xml but no longer has a wav file in the current directory. A blue circle means the length of a sound has been altered since the last time the Sound Volume Editor updated the sound.xml file. This adds the sound to the sound.xml file and makes it available to use in GR. To make a sound appear in the Available Elements list, there must be a check next to "Ambient Sound".

Other sound properties can also be altered here, such as the Volume and the 3d sound attributes under Volume. The "Test Sound at Position" isn't fully implemented and should not be used.

Note: the Sound Volume Editor should not be used while a mission has been loaded into Igor. It should only be used right after Igor is first run and before a mission has been loaded, or Igor may crash.

Vehicle Editor



The Vehicle Editor lets the user adjust the attributes of an existing vcl file or create a new one. Weapons can be added or removed (if the vehicle's model includes the correct data to support vehicles), and all of the physics can be adjusted, ranging from the weight of the vehicle to the speed at which it turns. Keep in mind any change made to an existing vehicle file will most likely break the scripting for that Vehicle in any missions where it's currently used.

Test Mission

This mission is a complete mission that you should use as a test of your Igor scripting ability. While it doesn't include every aspect of what a single-player mission can be in Ghost Recon, it is rather complex and should put most of what you've learned by reading this document to use. This will be a standard Ghost Recon mission, with three standard objectives and one special objective, three difficulty levels, multiplayer zones, a large enemy force, vehicles, and bombs.

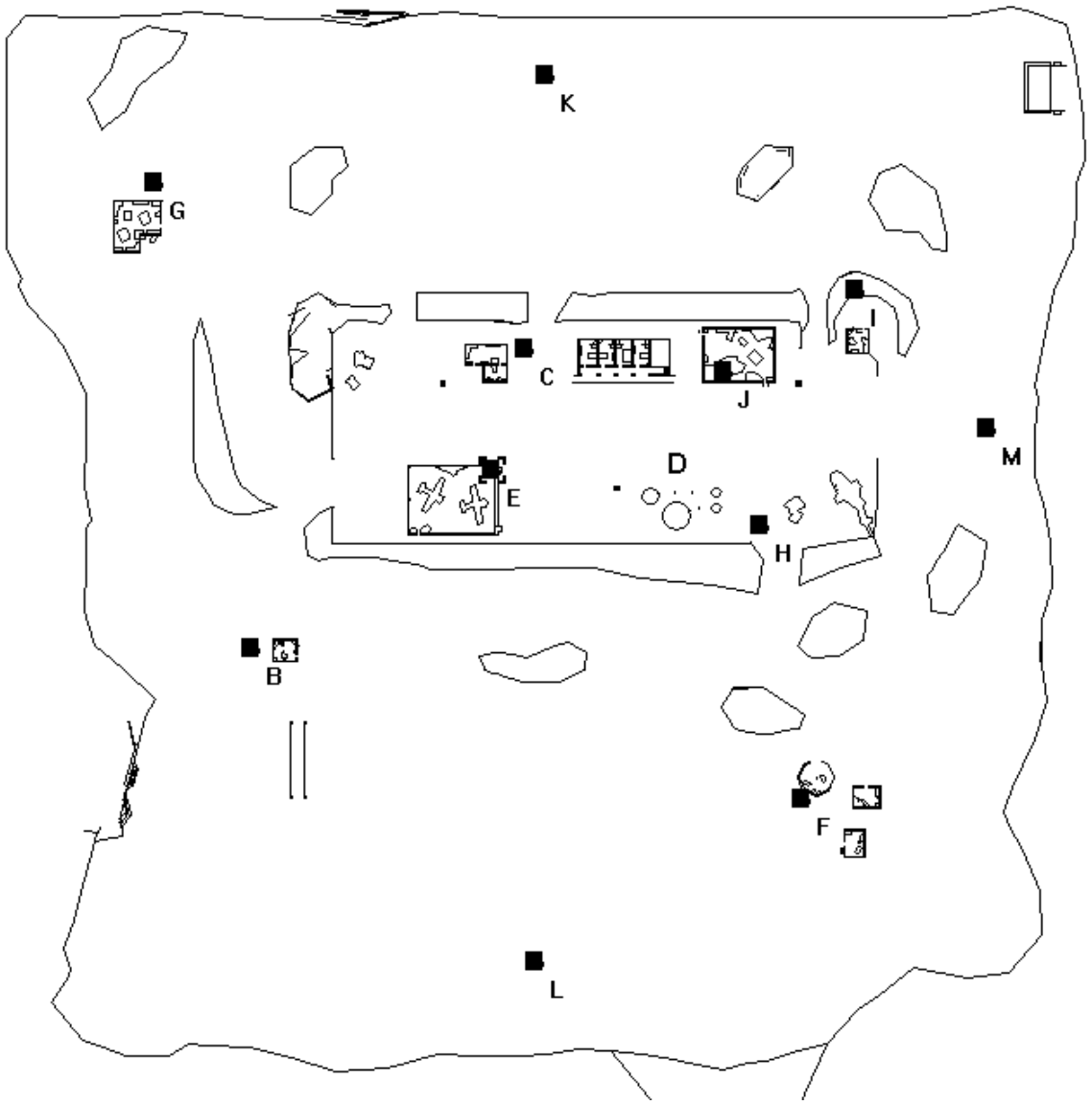
While you are scripting this mission, remember to save often and check your progress in-game as much as you can. The following console cheats may be helpful while you test (to activate the console, hit Enter on the number pad. Type in the cheat and press Enter to activate it, and press Esc to close the console):

Superman	-	Makes the player-controlled Actor invincible
Team Superman	-	Makes the player-controlled Platoon invincible
Shadow	-	Makes the player-controlled Actor undetectable, but they will still react to noise
Team Shadow	-	Makes all Actors undetectable by all other Actors -- this includes enemies
Run	-	Increases Player movement speed
Names	-	Shows every Actor's name from Igor above their heads
Aiinfo	-	Only works when Names is activated – shows what each Actor is doing beside his name
Cisco	-	Kills all enemies on the map

Keep in mind that cheats can and will 'break' the game. For example, if you have superman activated, testing the difficulty of a firefight is not recommended. Any mission should be tested with cheats on when necessary, and off to test for playability. Often, there are parts of the game you'll want to have cheats turned on for, like if you want to test a Zone that triggers some behavior when the player enters it, it helps to have run, superman, and shadow turned on so you can immediately run to and test just that Zone without having to worry about enemy Actors killing you along the way.

The tutorial mission will use the mission 4 swamp airfield map from Island Thunder. Special thanks to Christian Allen for the mission design doc. There is a summary of the script blocks used in the mission following the mission description, but try to not use it; you'll learn more if you figure things out for yourself.

P01 - Practice Mission



Map Name: Swamp Airfield
Mission Name: Jade Panther
Location: Lago Puraquequara, Brazil
Date: 10/15/04
Time: 11:00
Weather: Foggy

Background

The Brazilian government has requested American support in capturing a major drug cartel member, Gonzalo Rodriguez Gacha. Gacha is a major player in the Columbian drug trade, and due to recent CIA crackdowns in Columbia, has moved his operations to the deep Amazon in Brazil. While his whereabouts are unknown, the Brazilian authorities do have the location of one of his airstrips. A member of the Gacha cartel, Jesus Sabaneta, has offered to turn over Gacha's secure radio codes so that a fake radio message can be sent to Gacha to lure him out into the open, where he can be captured. Unfortunately, the local cartel members have learned that Sabaneta has been selling information, and is sending a group of gunmen to kill him. Sabaneta has one bodyguard to protect him, but they won't hold out long. The Ghosts must rescue Sabaneta from the gunmen and learn the secure radio codes. They must then fight their way into the airfield base, making sure that none of the cartel members send out any warning messages, and send the coded message out using the cartel's encrypted radio. Finally, the Ghosts need to plant explosives to blow the airfield, covering their involvement and cutting off a major supply site for Gacha.

Objectives:

1. Meet with Jesus Sabaneta
2. Send Radio Message
3. Plant Explosives at Fuel Tanks

X - Capture Airfield Commander

Briefing:

The Brazilian government has asked for our help in an anti-drug operation. They've come up with a plan to lure notorious cartel leader Gonzalo Rodriguez Gacha into a trap, and want to avoid any possible intelligence leaks that may come from using their own military. A member of the Gacha cartel, Jesus Sabaneta, has offered to give us the secure radio codes that the cartel uses on it's encrypted radio system. Once we have those codes, we can send a fake message over the cartel's own radio system, hopefully convincing Gacha into a meet.

Sabeneta is holed up in village just southeast of a Gacha airstrip. Gacha gunmen are on their way to take him out, so link up and protect him and his bodyguard, and then retrieve the radio codes from him. Then secure the airfield, send the coded radio message on the airfield's radio set, and then plant explosives on the large fuel tanks on the south side of the base to blow the whole place. Make sure that the airfield guards don't get a chance to send out a warning message, or the mission will be for nothing. If possible, capture the

airfield commander; he will be a valuable source of intel on the cartel's smuggling methods.
The terrain you are going into is swampy, and visibility will be poor. The cartel maintains several patrols of gunmen in the jungle around the airfield, so keep alert. Be careful out there, and make sure that all of your people are clear of the base when those fuel tanks blow!

Encounter Areas:

A-Insertion zone, on a sloping beach.
B-Small outpost area and rope bridge. One enemy mans an emplaced .50 cal (not placed in recruit), facing south. Four other enemies (2 in recruit, 3 in Veteran) are lounging inside the small building, and will rush out and defend the area if combat occurs nearby.
C- The Radio Building and a small barracks. Two enemies patrol around the radio building (1 in recruit), and a radio operator lounges in the barracks. If the patrol is engaged, they will fall back and defend the radio building, while the radio man will try to reach the radio and send a mayday message.
D- A large set of fuel tanks
E- A hanger with two civilian planes inside. Two enemies (1 on recruit) lounge inside this area. If the patrol in area C is engaged, these troops will rush to defend the radio building.
F- A small group of huts where Sabaneta is holed up. Sabeneta is in the southernmost hut, armed with a pistol, while his body guard is in the northwest hut, covering to the north. Both of these characters will be invulnerable until the player sees the enemy troops approaching from Area H.
G-A small warehouse. A large reserve force of seven enemies (1 on recruit, 4 on veteran) is quartered here, along with a large truck (not placed in recruit). Once the force around the radio building (Area C) is engaged, they will mount the truck and drive to the airfield, where they will dismount and set up defensive positions in the center of the field.
H-The southern entrance to the airfield. A group of eight enemies (4 on recruit) are here, and will move south to engage the body guard at Area F once the player passes east of Area B. Half of the force here is armed with LMG's. These troops will be invulnerable and engaged in a firefight with the bodyguard until the player sees them.
I- Airfield Command Post. This building contains the unarmed base commander, and two emplaced weapons, a machine gun and a grenade launcher (not placed in recruit), facing northeast. There is also one enemy (zero in recruit) patrolling around the building.
J- Warehouse. Two enemies (one in recruit) patrol the outside of this building.
K- North shore of island. Three enemies (2 on recruit) patrol from east to west in this area.
L- South shore of island. Three enemies (2 on recruit) patrol this area from east to west.
M- West shore of island. Three enemies (2 on recruit) patrol this area from south to north.

	Recruit (Easy):	Veteran (Medium):	Elite (Hard):
Total Actors:	23	38	42
Friendlies:	2	2	2
Enemies:	21	36	40
Neutrals:	0	0	0
Total Vehicles:	0	1	1
Friendly:	0	0	0
Enemy:	0	1	1
Neutral:	0	0	0

1. Meet with Jesus Sabaneta
    Kill all enemies attacking Sabeneta, then touch Sabeneta. Do not let him die before this.
2. Send Radio Message
    Touch Radio in Area C. Do not plant explosives first.
3. Plant Explosives at Fuel Tanks
    Plant demo charge at tanks in Area C. Do not run out of demo charges first. All actors within the area of the airfield will be killed if they are still there when the demo charges explode (30 seconds after set).

To unlock the hero the special objective must be achieved:
X - Capture Airfield Commander
    Touch Airfield Commander in Area I. Do not let him die.

Alternatively, if the player kills or routes all enemies on the map, he will complete the first three objectives and win the mission.

Failure Conditions

Sabaneta is killed before the player retrieves the radio codes.
The radioman in area C successfully sends a warning message (stands near radio for 5 seconds).
Team members are all killed before objectives are completed.
The player plants the demo charges at the fuel tanks before the radio message is sent.
The player runs out of demo charges the fuel tanks are destroyed.

Test Mission Script Summary

Group: <Default>
Comment:
  Initialize Player Platoon
Trigger Event:
  The simulation is starting.
Responses:
  Set Player Platoon to (The player-controlled platoon).
  Activate Debris Loading.
  Activate Explosion Loading.

Group: <Default>
Comment:
  Objective 1 Timer
Trigger Event:
  Objective 1 has expired.
Responses:
  Mark Objective 1 complete in the objective list.
  Increment Objective Counter.

Group: <Default>
Comment:
  Initialize Objective Counter
Trigger Event:
  1 second(s) elapsed.
Responses:
  Allow this block to be reactivated.
  Continue executing responses if ((The value of Objective Counter) is equal to 3).
  Display "Mission Complete" and register mission completion.

Group: <Default>
Comment:
  Objective 2 Timer
Trigger Event:
  Obective 2 has expired.
Responses:
  Mark Objective 2 complete in the objective list.
  Increment Objective Counter.

Group: <Default>
Comment:
  Objective 3 Timer
Trigger Event:
  Objective 3 has expired.

Responses:
  Mark Objective 3 complete in the objective list.
  Increment Objective Counter.

Group: <Default>
Comment:
  Objective X Timer
Trigger Event:
  Objective X has expired.
Responses:
  Mark Objective 4 complete in the objective list.
  Unlock the next hero character in the campaign.

Group: <Default>
Comment:
  Trigger Plan Area B Force
Trigger Event:
  BG1 has been killed.
Responses:
  Cancel execution of any plan assigned to Team - Area B Force.
  Assign Trigger Plan Area B Force to Team - Area B Force and execute.
  Cancel execution of any plan assigned to Team - Area B Gunner 2.
  Assign Trigger Plan Area B Gunner 2 to Team - Area B Gunner 2 and execute.

Group: <Default>
Comment:
  Firefight Invunerable
Trigger Event:
  1 second(s) elapsed.
Responses:
  Make all members of Platoon - Area F invincible.
  Make all members of Platoon - Area H invincible.

Group: <Default>
Comment:
  Route All Enemy
Trigger Event:
  All actors in Enemy have fled or been killed.
Responses:
  Set Objective 1 to expire in 1 second(s).
  Set Obective 2 to expire in 1 second(s).
  Set Objective 3 to expire in 1 second(s).
  Display "Mission Complete" and register mission completion.

Group: <Default>
Comment:

Area H Assault Trigger
Trigger Event:
A member of Player Platoon is within 10 meter(s) of Trigger Zone H Assault.
Responses:
Cancel execution of any plan assigned to Team - Area HA.
Cancel execution of any plan assigned to Team - Area HB.
Assign Trigger Plan HA Assault to Team - Area HA and execute.
Assign Trigger Plan HB Assault to Team - Area HB and execute.
Make FI1 vulnerable.

Group: <Default>
Comment:
Failure Intel Operative Killed
Trigger Event:
FI1 has been killed.
Responses:
Continue executing responses if ((The value of Objective Counter) is less than 1).
Mark Objective 1 failed in the objective list.
Display "Intel Operative Killed" and register mission failure.

Group: <Default>
Comment:
Area C Member Killed
Trigger Event:
A member of Team - Area C Defend has been killed.
Responses:
Continue executing responses if ((The value of Radio Trigger Counter) is less than 1).
Increment Radio Trigger Counter.
Cancel execution of any plan assigned to Team - Area C Defend.
Assign Plan - Area C Defend to Team - Area C Defend and execute.
Cancel execution of any plan assigned to Team - Area C Radio.
Assign Trigger Plan Area C Radio to Team - Area C Radio and execute.
Cancel execution of any plan assigned to Team - Area K.
Assign Trigger Plan Area K to Team - Area K and execute.
Cancel execution of any plan assigned to Team - Area M.
Assign Trigger Plan Area M to Team - Area M and execute.

Group: <Default>
Comment:
Failure Condition Radio Man Uses Radio
Trigger Event:
A member of Team - Area C Radio is within 10 meter(s) of Trigger Zone Radio.
Responses:
Set Radio Timer to expire in 5 second(s).

Group: <Default>

Comment:
  Radio Used Countdown
Trigger Event:
  Radio Timer has expired.
Responses:
  Continue executing responses if (CR1 is active).
  Display "Enemy Sent Transmission" and register mission failure.
  Mark Objective 2 failed in the objective list.

Group: <Default>
Comment:
  Trigger Zone Main Base Elite
Trigger Event:
  A member of Player Platoon is within 10 meter(s) of Trigger Zone Main Base.
Responses:
  Continue executing responses if (The campaign difficulty is set to Hard).
  Set Trigger Zone Main Base to expire in 10 second(s).
  Set Vehicle Assault Timer to expire in 15 second(s).

Group: <Default>
Comment:
  Trigger Zone Main Base Timer Expired
Trigger Event:
  Trigger Zone Main Base has expired.
Responses:
  Continue executing responses if ((The value of Radio Trigger Counter) is less than 1).
  Increment Radio Trigger Counter.
  Cancel execution of any plan assigned to Team - Area C Defend.
  Assign Trigger Plan Area C Defend to Team - Area C Defend and execute.
  Cancel execution of any plan assigned to Team - Area C Radio.
  Assign Trigger Plan Area C Radio to Team - Area C Radio and execute.

Group: <Default>
Comment:
  Team E Assault
Trigger Event:
  CR1 has been killed.
Responses:
  Cancel execution of any plan assigned to Team - Area E.
  Assign Trigger Plan E Assault to Team - Area E and execute.

Group: <Default>
Comment:
  Vehicle Assualt
Trigger Event:
  Vehicle Assault Timer has expired.

Responses:
  Assign Trigger Plan Vehicle to Truck - Area G and execute.

Group: <Default>
Comment:
  Vehicle Unload
Trigger Event:
  Truck - Area G is within 10 meter(s) of Trigger Zone Vehicle Destination.
Responses:
  Assign Trigger Plan Unload Vehicle to Team - Area G Driver and execute.
  Assign Trigger Plan Unload Vehicle to Team - Area G Force 1 and execute.
  Assign Trigger Plan Unload Vehicle to Team - Area G Force 2 and execute.

Group: <Default>
Comment:
  Demo Charge Placed
Trigger Event:
  A demo charge was placed within 10 meter(s) of Demo Charge Zone.
Responses:
  Set Demo Charge Placed to true.
  Set Explosion Timer to expire in 30 second(s).
  Set Explosion Kill Timer to expire in 30 second(s).
  Display "Charges set! Get out of there!" to all players for 3 second(s).

Group: <Default>
Comment:
  Explosion Timer 1 Expired
Trigger Event:
  Explosion Timer has expired.
Responses:
  Allow this block to be reactivated.
  Play "a_exmetal.wav" at volume 1.
  Activate Explosion 01.
  Activate Explosion 02.
  Activate Explosion 03.
  Activate Explosion 04.
  Activate Explosion 05.
  Activate Explosion 06.
  Activate Explosion 07.
  Activate Explosion 08.
  Set Explosion 2 to expire in 1 second(s).
  Skip the remaining responses if (The state of Debris 01 Fired).
  Activate Debris 01.
  Set Debris 01 Fired to true.

Group: <Default>

Comment:
  Fail if Obj 3 is completed First
Trigger Event:
  Explosion 2 has expired.
Responses:
  Continue executing responses if ((The value of Objective Counter) is less than 2).
  Mark Objective 2 failed in the objective list.
  Set Objective 3 to expire in 1 second(s).
  Display "You did not send the coded message!" and register mission failure.

Group: <Default>
Comment:
  Objective 1 Complete
Trigger Event:
  A member of Player Platoon is within 3 meter(s) of FI1.
Responses:
  Allow this block to be reactivated.
  Continue executing responses if ((The number of active actors on Platoon - Area H) is less than 1).
  Display "You have the radio codes." to all players for 3 second(s).
  Set Objective 1 to expire in 3 second(s).

Group: <Default>
Comment:
  Radio Message Sent
Trigger Event:
  A member of Player Platoon is within 10 meter(s) of Trigger Zone Radio.
Responses:
  Allow this block to be reactivated.
  Continue executing responses if ((The value of Objective Counter) is equal to 1).
  Display "Coded message sent." to all players for 3 second(s).
  Set Obective 2 to expire in 1 second(s).

Group: <Default>
Comment:
  Explosion Timer 2 Expired
Trigger Event:
  Explosion 2 has expired.
Responses:
  Allow this block to be reactivated.
  Activate Explosion 09.
  Activate Explosion 10.
  Activate Explosion 11.
  Activate Explosion 12.
  Activate Explosion 13.
  Activate Explosion 14.

Activate Explosion 15.
Activate Explosion 16.
Activate Explosion 17.
Activate Explosion 18.
Activate Explosion 19.
Activate Explosion 20.
Activate Explosion 21.
Activate Explosion 22.
Activate Explosion 23.
Activate Explosion 24.
Activate Explosion 25.
Activate Explosion 26.
Activate Explosion 27.
Activate Explosion 28.
Activate Debris 02.
Activate Debris 03.
Play "a_exmetal.wav" at volume 1.
Set Explosion Timer to expire in 1 second(s).
Continue executing responses if ((The value of Objective Counter) is equal to 2).
Set Objective 3 to expire in 2 second(s).
Continue executing responses if (Team - Area N Commander is currently being escorted).
Set Objective X to expire in 1 second(s).

Group: <Default>
Comment:
Commander Captive Mode
Trigger Event:
The simulation is starting.
Responses:
Set NC1 to captive behavior.

Group: <Default>
Comment:
Captive Killed
Trigger Event:
NC1 has been killed.
Responses:
Mark Objective 4 failed in the objective list.

Group: <Default>
Comment:
Initialize Demo Counter
Trigger Event:
1 second(s) elapsed.
Responses:

Allow this block to be reactivated.
Continue executing responses if ((The number of demo charges available to Player Platoon) is less than 1).
Skip the remaining responses if (The state of Demo Charge Placed).
Display ""Out of Demo Charges"" and register mission failure.

Group: <Default>
Comment:
Recruit Hide Weapons
Trigger Event:
The simulation is starting.
Responses:
Continue executing responses if (The campaign difficulty is set to Easy).
Hide Fixed GL - Area I from the game world.
Hide Fixed MG- Area B from the game world.

Group: <Default>
Comment:
Trigger Zone Main Base Veteran
Trigger Event:
A member of Player Platoon is within 10 meter(s) of Trigger Zone Main Base.
Responses:
Continue executing responses if (The campaign difficulty is set to Normal).
Set Trigger Zone Main Base to expire in 20 second(s).
Set Vehicle Assault Timer to expire in 30 second(s).

Group: <Default>
Comment:
Tigger Zone Main Base Recruit
Trigger Event:
A member of Player Platoon is within 10 meter(s) of Trigger Zone Main Base.
Responses:
Continue executing responses if (The campaign difficulty is set to Easy).
Set Trigger Zone Main Base to expire in 40 second(s).

Group: <Default>
Comment:
Firefight Vulnerable
Trigger Event:
1 second(s) elapsed.
Responses:
Allow this block to be reactivated.
Continue executing responses if ((A member of Platoon - Area H is being seen by Player Platoon) or ((The number of members of Player Platoon within 0 meter(s) of Trigger Zone Area H Vulnerable) is greater than 0)).
Make all actors in Allies vulnerable.

Make all actors in Enemy vulnerable.
Prevent this block from being reactivated.

Group: <Default>
Comment:
Explosion Death Kill Expired
Trigger Event:
Explosion Kill Timer has expired.
Responses:
Begin Cinematic Mode.
Place the camera at Camera A Position, facing Explosion 05.
Place the camera at Camera B Position, facing Explosion 15.
Call Base Commander Explosion Kill after this block.
Use Player Platoon Explosion Kill to loop over all actors in Player Platoon after this block.
Use Enemy Explosion Kill to loop over all actors in Enemy after this block.

Group: <Default>
Comment:
Trigger Area L Move to F
Trigger Event:
A member of Platoon - Area H has been killed.
Responses:
Cancel execution of any plan assigned to Team - Area L.
Assign Trigger Plan Area L to Team - Area L and execute.

Group: Base Commander Explosion Kill
Comment:
Base Commander Explosion Kill
Trigger Event:
This block has been called directly from the script.
Responses:
Continue executing responses if ((The number of members of Team - Area N Commander within 10 meter(s) of Trigger Zone Main Base) is greater than 0).
Have NC1 commit suicide.

Group: Enemy Explosion Kill
Comment:
Death Kill Enemy Company
Trigger Event:
An actor loop is ready to process This Actor.
Responses:
Allow this block to be reactivated.
Continue executing responses if ((The number of members of (The team including This Actor) within 0 meter(s) of Trigger Zone Main Base) is greater than 0).
Have This Actor commit suicide.

Group: Player Platoon Explosion Kill
Comment:
  Loop Platoon Members Explosion Kill
Trigger Event:
  An actor loop is ready to process This Actor.
Responses:
  Allow this block to be reactivated.
  Continue executing responses if ((The number of members of (The team including This Actor) within 0 meter(s) of Trigger Zone Main Base) is greater than 0).
  Have This Actor commit suicide.

7: In-Depth Multiplayer Scripting

Multiplayer scripting takes the linear scripting of single player and turns it on its ear. Little of the scripting in multiplayer is linear, and becoming used to the shift in scripting logic can be difficult. Multiplayer scripting is largely comprised of loops and repeating blocks that do not specify anything in the Placed Elements List, but instead specify References for Elements. Think of References as an empty pool of a certain kind of Element. When you create a Reference, you are preparing one of these empty Elements to be defined and used in the script.

Multiplayer scripting isn't saved in a mission file – it's saved in a GTF, which stands for Game Type File. When scripting a GTF, no map should be loaded into Igor. All loading and saving is done via the Script Menu in the Menu Bar. To load a GTF, click on Import and to save click on Export. When you Import a GTF, it will delete any script that is currently loaded.

Although a few different aspects of multiplayer scripting are detailed below, the main focus of this section is the tutorial. Since multiplayer scripting is mainly an exercise in circular logic and loops, once the user takes the leap of logic from single-player to multiplayer scripting the syntax and process take a back seat to problem solving.

Objectives are saved to the script, not the mission file, so if objectives are added to a GTF and the GTF is exported, the Objectives are exported in the GTF, too.

Loops

GTFs rely on loops. Since no Element can be placed in a GTF, the script must rely on assigning blank References the values a placed Element would normally have. Refs are covered below. When handling multiple Element References, the easiest way to do this is through a loop.

Loops are specific sets of Triggers and Responses organized through Groups. Although users can set up a basic "loop" using any block that's preserved and a reoccurring Trigger, real loops use the "QueueLoop" Responses and the "Loop" Triggers. For example, let's say the user wanted to assign every Team in a Platoon the exact same Plan. This can be done without loops, but it would be an unnecessary exercise in tedium, especially if that Platoon contained lots of Teams.

In this example, let's say the user sets up one block (regardless of the block's Trigger) with the QueueLoopTeamsinPlatoon Response in there somewhere (which specified the "Loop A" Group along with the appropriate Platoon) and a block in the "Loop A Group" Triggered by LoopTeams. In that Trigger, the user creates a TeamRef (see below) called "This Team", so the Trigger activates when a loop is ready to process "This Team". The block needs to be preserved so it happens multiple times, and the only response is "ExecutePlanTeam". If the Plan is named "Plan – Run", the Response is set up so that Plan – Run is assigned to "This Team", which the first time through the loop is automatically the first Team in that Platoon. The next time the loop runs, which will be immediately after the first time, "This Team" will be the second Team in that Platoon, the third time "This Team" will automatically be the third Team in the Platoon, etc… until the Platoon has no more Teams left in it, at which point the loop automatically ends.

Loops can be confusing because the user doesn't tell the loop block itself to happen again, because loops that don't require the use of "Loop" Triggers or Responses do require the user to make sure the block repeats. The "QueueLoop" Responses tell the block under the designated Group to keep happening until it runs out of Elements to loop through.

Refs

Assigning value to these References requires the use of a generic Reference of the category the Element falls into. In Igor, most Elements that can be placed in a mission can also have a blank Reference created for them. Therefore, in the Tags section of the Script Editing window there are not only Elements like Zones, Actors, etc., but also ZoneRefs, ActorRefs, and Refs for most other Elements. A GTF never needs to use anything except Refs if there's the option between a Ref and a non-Ref Element. Use Refs in GTFs.

Multiplayer GTF Scripting Tutorial

In this tutorial, we'll be making a completely new GTF for a new multiplayer mode – Capture Papashvili. Before we start, you should have a pretty good grasp of Igor's functionality and single-player scripting, and you should have looked through earlier sections on multiplayer scripting. This tutorial will be a step-by-step dissection of the process used to create a GTF from scratch and an explanation to the logic behind each step. It is recommended that you have some paper and a printer handy, as you'll need to have some scratch paper to jot down sudden bursts of inspiration and a printer to print out the summary of the GTF as we go along. Often times having a hard copy of the script summary in front of you and leaving the computer to look it over can help fix any errors in your scripting. Remember to comment these sections well, or else you'll get lost in the scripting easily.

The "Capture Papashvili" is a Team-based game mode. It is about leading Papashvili from the Central Area and back to your Team's base. When he is escorted to a Team's base that Team receives a point and Papashvili teleports back to the Central Area. Whatever Team has the most points when everyone is dead or the time runs out wins. Sounds simple, right?

The first thing we have to do is make sure Igor isn't running a specific mission file. Just close Igor and reopen it without loading a mission. Then open up the Script Editing window. Come up with a naming convention for the comment line for each block. If things are going to happen in order or at specific times, add numbers to the front of each block comment that you make to help you keep track of everything. Make sure the number is two-digit, so that numbers under ten will sort properly.

Here's the things we know we'll have to do for all this to work:

- Spawn Papashvili
- Set up the bases to show up on the command map to the company for each different player Team and make the zone emit smoke
- Keep track of each Team's score
- Compare each Team's score to figure out if there's going to be a winner (and thus losers) or a draw when everyone dies or when time runs out
- Set up Papashvili so he's able to be captured
- Teleport Papashvili back to the Central Area when he's brought back to a Team's base
- Have Papashvili be able to be captured if the escorting Team is killed

Some of these things will be in separate blocks, and others will be in the same blocks, and some will be stretched over multiple blocks. First, though, you'll have to create an Objective by going to "Script, Objectives" called "Capture Papashvili". After that, we'll start with spawning Papashvili through the Scripting Language.

For an Actor to be spawned into a multiplayer game all of the supporting organization (Team, Platoon, and Company) must be in place first. Since spawning **must** be done in a block triggered by PreAction, make a new block in the <default> Group triggered by the PreAction Trigger. Now we start to spawn in the supporting organization. First, we'll use the SpawnCompany Response to create his company.

Click “Edit…” in the Tag section and you’ll see that “CompanyRef” is the default Element category. Enter “Captive Company” and press the Apply button. Now, select Captive Company from the pull-down menu next to “Tag”. Do this for Captive Platoon and Captive Team.

When you spawn Papashvili, you’ll have to enter filenames into the Literal Parameter. Put in “m01_papashvili.atr” and “NoWeapon.kit” verbatim into that Parameter. If you don’t, you’ll make the game crash when you try and use this GTF. With all this done, that’s the first script block complete for the GTF.

The next block will initialize some necessary parts of the GTF and set up some loops that make sure the appropriate players have the correct Zones assigned to them. It is still part of the <default> group, and the trigger is StartUp. Since we don’t want the game to automatically end without lettings us calculate the score to determine a winner at the end, use the “DisableAutoEnd…” Responses so it won’t end when the game timer expires or when there are no players left. We don’t want anyone killing Papashvili, so next we’ll set everyone on his Company to be invincible and invulnerable. Then, set all the members of his Team to Captive, and show him on the command map to all the players.

Now, we’re going to set up some loops to determine what player Teams get what Bases. Since there are 4 bases, we’ll have to set up 4 loops. Use the “QueueLoopCompanies” Response, and set one up to use a new Group, “Loop for Base 0”. The next three will use “Loop for Base 1” and so on. Remember, anything Queued will happen after the current block is finished running. We still have some things to set up, so now use the Variable Set Zone to set a ZoneRef for each Base to the actual Base (using Queries). Some of the Responses we’ll be using later won’t be able to Query for Bases, so we’ll need these refs. And speaking of things we’ll be using later, go ahead and make a Flag called “Flag – Ending” and set it to false, a Counter called “Counter – Number of Leaders” and set it to 0, and a Counter called “Counter – High Score” and set that to 0 also. We’ll need these before too long.

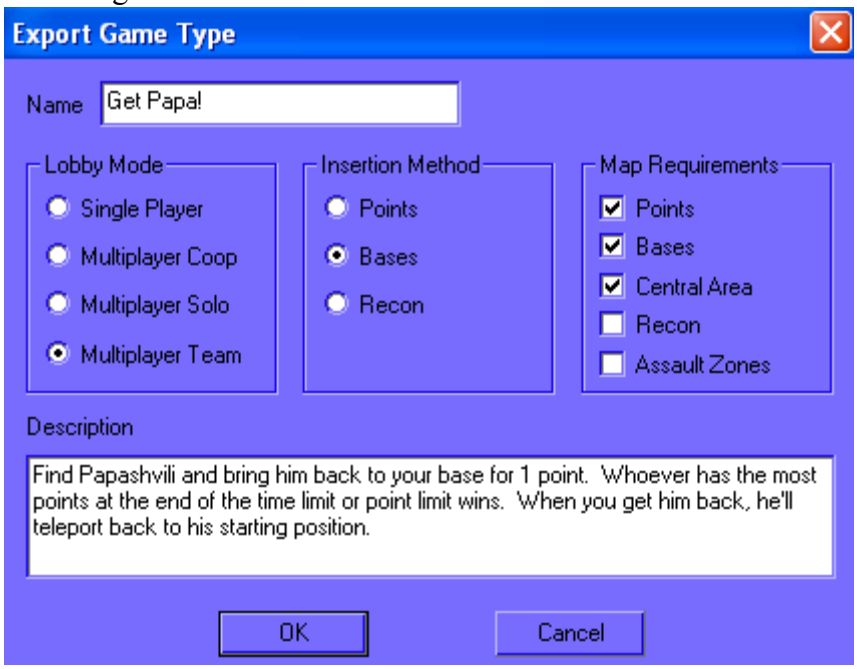
The next four blocks deal with the loops we set up in the previous block. Each one is Triggered by the LoopCompanies response, and as stated in the previous part of this section you’ll have to use “This Company” as the Company referred to in the Trigger. Each one does the same thing, but for a different Base. Make sure each one is in the appropriate and different Group (“Loop for Base 0”, and so on). First thing’s first – make sure this block is preserved with a “BlockPreserve” Response from the get go. Now, we’ll set up a ContinueIf to check to see how close members of “This Company” are to the Base we’re checking for in this block. Right now, that’s Base 0. It’ll continue if the number of members of “This Company” within 0 meters of Base 0 is greater than 0. If that’s true, then we want to set Company 0 (a CompanyRef) to “This Company”, show Base 0 to all members of Company 0 and have it smoke, enable a group called “Drop Guy off at Base 0”, and then use a BlockRemove to keep this block from being activated again. For the next block, the setup is the same but instead of Base 0, Company 0, and “Drop Guy off at Base 0” we’ll change those 0’s to 1’s. Do this again for all four Companies, 0-3.

Much like those loops, the setup for the next four blocks is the same, but they reference different Bases and different Companies. These blocks are part of the Groups you enabled in the previous loops. The first one is part of the “Drop Guy off at Base 0”

Group. Use a BlockPreserve first, and then get ready for a convoluted ContinueIf. First, use ContinueIf, and make it use "Or". Then, for the first part, find out if the "inverse of (the company currently escorting Captive Team) is a valid reference" – I.E. if they're playing the game. For the 2nd part, find out if the "Company currently escorting Captive Team is equal to Company 0" using a CompareThings Query.

After that, turn off the Team AI for Captive Team, Display a message letting all members of Company 0 letting them know that Papashvili's been captured, display another message to everyone that Papashvili's been returned to the central area, and maybe play a sound that people will identify with that happening. Then, teleport all members of Captive Platoon to the Central Area using "TeleportPlatoontoZone". Turn on the Team AI for the Captive Team, set Captive Team to the captive behavior, and increment the Score for Company 0. Repeat this for Company 1-3, with each in their own block. Remember to set the Group correctly for each one.

If you were to export the GTF now (probably a good thing to do considering all that we've done so far), you'll get a screen where you can type in its name and then the following screen:



You can type in the name that will appear in the multiplayer server setup screen, what kind of game it'll be in the lobby, how people will insert, and what zones are required for the GTF to be played. You can also put in a description that will appear when a player looks at the briefing for the GTF. Go ahead and fill it in with the same information that you see in this screenshot, and click "ok".

Now, if you were to go into GR and start a multiplayer game using this GTF, it should successfully start, you should be able to go to Papashvili, bring him back to your base, see him teleport back to the central area after you score a point, and be able to capture him over again. Next, we'll set up the GTF to figure out who is winning and losing the game. Remember that when loading a GTF, you don't say "File, Load" – you go to "Script, Import".

The next block will end the game and set things in motion to figure out who wins, loses, or ties if the Game Timer expires. It's in the <default> Group, and triggered when the Game Timer expires. The first thing we need to do here is make sure that if the game is ending because there are no players left then this block doesn't do anything. Make a StopIf that checks to see if "Flag – Ending the Game" is true. After that, let everyone

know the game is ending, and then set "Flag – Ending the Game" to be true. Use QueueLoopCompanies to use a new Group called "Calculate High Score", and after that set up another loop using a new Group called "Determine Tie". Then go ahead and enable yet another new Group called "Determine Winner" with a simple GroupEnable Response. The next block is exactly the same as the previous one, except that instead of being triggered by the Game Timer expiring, it is triggered by all the players leaving the game.

The next block is part of the "Calculate High Score" Group, and is the loop that calculates who has the high score at the end of the game. The Trigger is LoopCompanies, and use "This Company" again. Use BlockPreserve and then a ContinueIf (using "And") that checks to see if a player on "This Company" is connected to the game and if the score for "This Company" is greater than or equal to the value of Counter – High Score. After that, set Counter – High Score to a Query for the score for "This Company". And then set a new CompanyRef, "High Score Company" to "This Company".

The next block is part of the "Determine Tie" Group, and is the loop that figures out if there's a tie. The Trigger is LoopCompanies, and use "This Company" yet again. Start off by using BlockPreserve. After that, make a ContinueIf to check to see if a player on "This Company" is connected. You can either make that ContinueIf use "And", or make another ContinueIf that checks to see if the score for "This Company" is equal to the value of Counter – High Score. If it is, increment Counter – Number of Leaders.

This new block is a 'gateway' block – it determines if there's a winner, loser, or a draw by running loops. It's part of the "Determine Winner" Group, and is triggered by 1 second elapsing. Since it's not part of the <default> Group, this 1 second elapses after the Group has been enabled. Just have this block do a call to the "Assign Winner" Group. If this GTF was more complicated, there could be other important responses that would happen here, too, and if you feel like making the GTF more complicated by all means do so, but after the tutorial.

In the next block, which is part of the "Assign Winner" Group, we'll see if anyone really won or lost the game. Its triggered by being called directly from another part of the script. First, end the game and display nothing using the "EndGameAll" Response. Don't worry about the timing – everything that needs to happen will happen before the game actually gets around to ending. Next, use a RedirectIf to see if the value of Counter – Number of Leaders is greater than 1 – if it is, call a new Group called "Make Tie Happen". After that, do a ContinueIf, and check to see if the value of Counter – Number of leaders is equal to 1. If it is, mark the "Capture Papashvili" Objective complete for the High Score Company. After that, mark the High Score Company as a winner and display "Victory!" to its members using the MarkCompanyWin Response. After that, queue a Company loop using a new Group – "Assign Losers".

Now, we'll tackle looking for a tie. This block is in the "Assign Tie" Group, and is triggered by CompanyLoop using "This Company". Preserve the block first, and then do a ContinueIf to check to see if the value of Counter – Number of Leaders is greater than 1 and if the value of Counter – High Score is equal to the score for "This Company". If so, then mark "This Company" as tied and display "The game has ended in a draw" to its members.

This block is in the “Assign Losers” Group and is triggered by Company Loop using “This Company”. First, preserve the block. Then, do a StopIf and check to see if “This Company” is equal to the High Score Company, because you don’t want the winners to lose. Next, mark “This Company” as a loser and display “Defeat!” to its members, and then mark the Capture Papashvili Objective as failed for members of “This Company”.

The next-to-final block is in the “Make Tie Happen” Group, and is triggered by being called from elsewhere in the script. All this block does is queue a company loop using the “Assign Tie” Group.

The last block is part of the <default> Group, and is here just for cleanup. If Papashvili’s escort is killed, we want to make sure that other players can steal him away. So the Trigger is EscortAborted and we let this block be reactivated. After that, turn off the Team AI for Captive Team, and then turn it back on. Finally, set the Captive Team to Captive. You have to shut off the AI off and turn it back on so it forgets who was escorting it last.

Now if you export it (go ahead and overwrite it if you exported it before) and load it up in GR, you’ll have your very own GTF that lets you capture Papashvili over and over again with your friends. Play around and experiment with the GTF and incorporate your own ideas into it. It’ll work just like official GTFs in that it’ll work with all maps that have a mission associated with them that include the required zones.

Congratulations and have fun making new GTFs!

Capture Papashvili GTF Script Summary

Group: <Default>
Comment:
  00: Spawn Captive Company & All Elements in that Company
Trigger Event:
  The simulation is initializing.
Responses:
  Spawn Captive Company.
  Spawn Captive Platoon onto Captive Company.
  Spawn Captive Team onto Captive Platoon.
  Spawn Captive Guy onto Captive Team with "m01_papashvili.atr" and "NoWeapon.kit" at (The central area).

Group: <Default>
Comment:
  01: Initialize
Trigger Event:
  The simulation is starting.
Responses:
  Do not end the game automatically when there are no players left.
  Do not end the game automatically when the game timer expires.
  Make all Actors in Captive Company invincible.
  Make all Actors in Captive Company undetectable.
  Set all members of Captive Team to captive behavior.
  Show Captive Guy on the command map for all players.
  Use Loop for Base 0 to loop over all companies after this block.
  Use Loop for Base 1 to loop over all companies after this block.
  Use Loop for Base 2 to loop over all companies after this block.
  Use Loop for Base 3 to loop over all companies after this block.
  Set Base 0 to (Base 0).
  Set Base 1 to (Base 1).
  Set Base 2 to (Base 2).
  Set Base 3 to (Base 3).
  Set Flag - Ending the Game to false.
  Set Counter - High Score to 0.
  Set Counter - Number of Leaders to 0.

Group: Loop for Base 0
Comment:
  02a: Loop - who is at Base 0
Trigger Event:
  A company loop is ready to process This Company.
Responses:
  Allow this block to be reactivated.

Continue executing responses if ((The number of members of This Company within 0 meter(s) of Base 0) is greater than 0).
Set Company 0 to This Company.
Show Base 0 on the command map for members of Company 0, highlight = true.
Enable the Drop Guy off at Base 0 script blocks.
Prevent this block from being reactivated.

Group: Loop for Base 1
Comment:
02b: Loop - who is at Base 1
Trigger Event:
A company loop is ready to process This Company.
Responses:
Allow this block to be reactivated.
Continue executing responses if ((The number of members of This Company within 0 meter(s) of Base 1) is greater than 0).
Set Company 1 to This Company.
Show Base 1 on the command map for members of Company 1, highlight = true.
Enable the Drop Guy off at Base 1 script blocks.
Prevent this block from being reactivated.

Group: Loop for Base 2
Comment:
02c: Loop - who is at Base 2
Trigger Event:
A company loop is ready to process This Company.
Responses:
Allow this block to be reactivated.
Continue executing responses if ((The number of members of This Company within 0 meter(s) of Base 2) is greater than 0).
Set Company 2 to This Company.
Show Base 2 on the command map for members of Company 2, highlight = true.
Enable the Drop Guy off at Base 2 script blocks.
Prevent this block from being reactivated.

Group: Loop for Base 3
Comment:
02d: Loop - who is at Base 3
Trigger Event:
A company loop is ready to process This Company.
Responses:
Allow this block to be reactivated.
Continue executing responses if ((The number of members of This Company within 0 meter(s) of Base 3) is greater than 0).
Set Company 3 to This Company.
Show Base 3 on the command map for members of Company 3, highlight = true.

Enable the Drop Guy off at Base 3 script blocks.
Prevent this block from being reactivated.

Group: Drop Guy off at Base 0
Comment:
03a: Drop Guy off at Base 0
Trigger Event:
An Actor in Captive Company is within 10 meter(s) of Base 0.
Responses:
Allow this block to be reactivated.
Continue executing responses if ((The inverse of ((The company currently escorting Captive Team) is a valid reference)) or ((The company currently escorting Captive Team) is equal to Company 0)).
Turn off Team AI for Captive Team.
Display "Papashvilli Captured!" to all members of Company 0.
Display "Papashvilli returned to Central Area" to all players.
Play "a_sirenwav" at volume 1.
Teleport all members of Captive Platoon to (The central area).
Turn on Team AI for Captive Team.
Set all members of Captive Team to captive behavior.
Increment the score for Company 0.

Group: Drop Guy off at Base 1
Comment:
03b: Drop Guy off at Base 1
Trigger Event:
An Actor in Captive Company is within 10 meter(s) of Base 1.
Responses:
Allow this block to be reactivated.
Continue executing responses if ((The inverse of ((The company currently escorting Captive Team) is a valid reference)) or ((The company currently escorting Captive Team) is equal to Company 1)).
Turn off Team AI for Captive Team.
Display "Papashvilli Captured!" to all members of Company 1.
Display "Papashvilli returned to Central Area" to all players.
Play "a_sirenwav" at volume 1.
Teleport all members of Captive Platoon to (The central area).
Turn on Team AI for Captive Team.
Set all members of Captive Team to captive behavior.
Increment the score for Company 1.

Group: Drop Guy off at Base 2
Comment:
03c: Drop Guy off at Base 2
Trigger Event:
An Actor in Captive Company is within 10 meter(s) of Base 2.

Responses:
  Allow this block to be reactivated.
  Continue executing responses if ((The inverse of ((The company currently escorting Captive Team) is a valid reference)) or ((The company currently escorting Captive Team) is equal to Company 2)).
  Turn off Team AI for Captive Team.
  Display "Papashvilli Captured!" to all members of Company 2.
  Display "Papashvilli Returned to Central Area" to all players.
  Play "a_sirenwav" at volume 1.
  Teleport all members of Captive Platoon to (The central area).
  Turn on Team AI for Captive Team.
  Set all members of Captive Team to captive behavior.
  Increment the score for Company 2.

Group: Drop Guy off at Base 3
Comment:
  03d: Drop Guy off at Base 3
Trigger Event:
  An Actor in Captive Company is within 10 meter(s) of Base 3.
Responses:
  Allow this block to be reactivated.
  Continue executing responses if ((The inverse of ((The company currently escorting Captive Team) is a valid reference)) or ((The company currently escorting Captive Team) is equal to Company 3)).
  Turn off Team AI for Captive Team.
  Display "Papashvilli Captured!" to all members of Company 3.
  Display "Papashvilli Returned to Central Area" to all players.
  Play "a_sirenwav" at volume 1.
  Teleport all members of Captive Platoon to (The central area).
  Turn on Team AI for Captive Team.
  Set all members of Captive Team to captive behavior.
  Increment the score for Company 3.

Group: <Default>
Comment:
  04a: End Condition = Game Timer Expired
Trigger Event:
  The game timer has expired.
Responses:
  Skip the remaining responses if (The state of Flag - Ending the Game).
  Display "Game Timer Expired" to all players.
  Set Flag - Ending the Game to true.
  Use Calculate High Score to loop over all companies after this block.
  Use Determine Tie to loop over all companies after this block.
  Enable the Determine Winner script blocks.

Group: <Default>
Comment:
  04b: End Condition = No Players Left
Trigger Event:
  All players have died or left the game.
Responses:
  Skip the remaining responses if (The state of Flag - Ending the Game).
  Set Flag - Ending the Game to true.
  Use Calculate High Score to loop over all companies after this block.
  Use Determine Tie to loop over all companies after this block.
  Enable the Determine Winner script blocks.

Group: Calculate High Score
Comment:
  05a: Calculate High Score
Trigger Event:
  A company loop is ready to process This Company.
Responses:
  Allow this block to be reactivated.
  Continue executing responses if ((A player on This Company is connected) and ((The score for This Company) is greater than or equal to (The value of Counter - High Score))).
  Set Counter - High Score to (The score for This Company).
  Set High Score Company to This Company.

Group: Determine Tie
Comment:
  05b: Check for Score Leaders
Trigger Event:
  A company loop is ready to process This Company.
Responses:
  Allow this block to be reactivated.
  Continue executing responses if (A player on This Company is connected).
  Continue executing responses if ((The score for This Company) is equal to (The value of Counter - High Score)).
  Increment Counter - Number of Leaders.

Group: Determine Winner
Comment:
  06: Determine Winner, tie, or loser
Trigger Event:
  1 second(s) elapsed.
Responses:
  Call Assign Winner after this block.

Group: Assign Winner
Comment:
  06a: Winner
Trigger Event:
  This block has been called directly from the script.
Responses:
  End the game and display ""
  If ((The value of Counter - Number of Leaders) is greater than 1), stop and queue a call to Make Tie happen.
  Continue executing responses if ((The value of Counter - Number of Leaders) is equal to 1).
  Mark Capture Papashvili complete in the objective list for High Score Company.
  Mark High Score Company as a winner and display "Victory!" to its members
  Use Assign Losers to loop over all companies after this block.

Group: Assign Tie
Comment:
  06b: Assign Tie
Trigger Event:
  A company loop is ready to process This Company.
Responses:
  Allow this block to be reactivated.
  Continue executing responses if ((The value of Counter - Number of Leaders) is greater than 1).
  Continue executing responses if ((The value of Counter - High Score) is equal to (The score for This Company)).
  Mark This Company as tied and display "The game has ended in a draw" to its members

Group: Assign Losers
Comment:
  06c: Losers
Trigger Event:
  A company loop is ready to process This Company.
Responses:
  Allow this block to be reactivated.
  Skip the remaining responses if (This Company is equal to High Score Company).
  Mark This Company as a loser and display "Defeat!" to its members
  Mark Capture Papashvili failed in the objective list for This Company.

Group: Make Tie happen
Comment:
  06d: Make Tie Happen
Trigger Event:
  This block has been called directly from the script.
Responses:

Use Assign Tie to loop over all companies after this block.

Group: <Default>
Comment:
07: Make sure Papa's captive if escort ends
Trigger Event:
The escort of Captive Guy has been aborted.
Responses:
Allow this block to be reactivated.
Turn off Team AI for Captive Team.
Turn on Team AI for Captive Team.
Set all members of Captive Team to captive behavior.

8: Appendix

The Appendix contains miscellaneous information and references for things that this document has covered, as well as new information that doesn't fit anywhere else.

Glossary

Element - An Element is the generic category for Actors, Companies, Teams, Zones, Weapons, etc. Anything that can be placed or referenced using the Scripting Language is an Element.

Actor - An Actor is any moving, AI-controlled or Player-controlled character that appears during the game. An Actor has equipment that is designated by his Kit, a name that Igor uses to reference the Actor, and an Actor's Actor File defines the statistics and graphics of any given Actor.

Team - A Team is a collection of no more than six Actors. Teams fall into a Platoon, and Actors. Actors within a Team make use of Team-level AI. A Team can be assigned a Plan.

Platoon - A Platoon is a collection of Teams. A Platoon can hold an unlimited number of Teams. Teams within a Platoon make use of Platoon-level AI.

Company - A Company is a collection of Platoons and Vehicles. There are usually only two Companies on any given mission, the Player-Allied Company and the Enemy Company.

Tag - A Tag is what classifies an Element as either an Actor, Zone, Company, Platoon, Vehicle, Weapon, etc…

Scripting - Scripting is the act of creating a mission using Igor. Scripting covers not only the Scripting Language in Igor, but also the placement of various Effects for single player and multiplayer gameplay, filling in Briefing information, assigning combat points for the mission, determining the default Platoon setup for the player, creating objectives, etc.

